SOTA Benutzer-Dokumentation - Version 1.0

1 Einführung

SOTA ist ein Tool für die statische Programmanalyse und den strukturorientierten Programmtest (**S**truktur**O**rientierter **T**est und **A**nalyse). Im Rahmen des strukturorientierten Programmtests ist es seine Aufgabe, die Überdeckung des Quellcodes beim Testen eines Programmes zu ermitteln, verschiedene Überdeckungsmaße dazu zu berechnen und diese Informationen visuell aufzubereiten. Die ermittelten Informationen lassen eine Bewertung der Güte eines Programmtests in Hinblick auf die Quellcodeabdeckung zu, nicht überdeckte Quellcodeabschnitte oder nicht ausreichend getestete Bedingungen lassen sich einfach identifizieren. SOTA ist also nicht direkt für den Test des Programmes verantwortlich, sondern dient als Hilfsmittel zur Bewertung von Testfällen und der Entwickelung ergänzender Testreihen.

SOTA ermittelt die Überdeckung durch Instrumentierung des Quellcodes, d.h. das zu testende Programm muss als kompilierbarer Quellcode vorliegen.

SOTA 1.0 arbeitet ausschließlich auf Java-Programmen, ist aber dafür ausgelegt, alle gängigen imperativen und objektorientierten Programmiersprachen unterstützen zu können. Um SOTA für andere Programmiersprachen einsetzen zu können, müssen ein Parser und verschiedene Klassen zur Abbildung der Programmiersprachenstruktur auf eine abstraktere Struktur zur Verfügung gestellt werden. Die Spezifikation dazu kann man in der Entwicklerdokumentation nachlesen.

Das Programm wurde als eine Standalone-Eclipse-RCP-Application entwickelt und läuft unter Windows2000 und aufwärts. Für den Einsatz in automatischen Testsystemen wird die Non-GUI-Funktionalität von SOTA durch die Bibliothek SOTA-ATM.jar bereitgestellt, welches auch eine API für die Integration in anderen Programmen bietet.

Kapitel 2 vermittelt einen Überblick über die Hauptfunktionen des Programmes und seine Nutzungsarten. Der Umgang mit dem Programm wird anschließend ausführlich in den Kapiteln 4-6 behandelt. Hierbei erfolgt in den Kapiteln 4 und 5 eine systematische und im Kapitel 6 eine prozessorientierte Beschreibung. Letztere beinhaltet typische Anwendungsszenarien in Form von Tutorien. Für ein besseres Verständnis werden abschließend verwendete Maße und sonstige Begriffe erläutert.

Das Programm dient ausschließlich dem Einsatz in der Lehre!

SOTA Benutzer-Dokumentation - Version 1.0

2 Überblick

SOTA unterstützt die statische Programmanalyse und den dynamischen Programmtest.

Bei der statischen Programmanalyse wird der Quellcode des Programmes analysiert und im Ergebnis werden durch SOTA zehn statische Maße wie die Zyklomatische Komplexität oder die Anzahl der Modifizierten Boundary-Interior-Pfade ermittelt. Voraussetzung für die Analyse ist das Parsen des Quellcodes durch SOTA.

Der dynamische Programmtest (nachfolgend Programmtest genannt) ist ein strukturorientiert-kontrollflussbezogener Programmtest und bildet den Hauptteil von SOTA. Abhängig vom speziellen Testfall ermittelt SOTA während der Programmabarbeitung neun Überdeckungsmaße wie z.B. die Zweigüberdeckung oder die Mehrfach-Bedingungsüberdeckung und stellt die Überdeckung graphisch dar. Voraussetzung hierfür ist eine Instrumentierung des zu testenden Programes durch SOTA.

Es gibt drei grundlegende Arten SOTA im Programmtest einzusetzen: der *manuelle Programmtest*, zusammen *mit einem externen Testsystem* oder *integriert in ein automatisches Testsystem*.

Im *manuellen Programmtest* wird das Programm manuell getestet, d.h. per Hand gestartet, Funktionen werden ausgeführt etc. Dies kann durch eine Entwicklungsumgebung wie Eclipse erfolgen, die neben SOTA benutzt wird, oder aus SOTA heraus unter Nutzung eines Ant-Buildfiles und eines Startskripts.

Die Arbeit mit einem *externen Testsystem* ist davon kaum verschieden, lediglich die Art des Testens wird davon betroffen. SOTA übernimmt unverändert, wie beim manuellen Test, die Vor- und Nachbereitung. Für den Programmtest selber wird jedoch ein separates Testsystem genutzt, wie z.B. ATOSj. Dabei ergibt sich eine Programmsequenz SOTA, ATOSj, SOTA und dies ohne jegliche interne Kopplung.

Bei der Arbeit mit einem *automatischen Testsystem* kann SOTA als Bibliothek (SOTA-ATM) in dieses System eingebunden und von dort die Non-GUI-Funktionalität von SOTA verwendet werden. SOTA-Kernfunktionen werden dann entweder über Kommandozeilenparameter oder an einer SotaATM-Instanz, deren Klasse durch die Bibliothek zur Verfügung gestell wird, aufgerufen.

Die Arbeit mit SOTA gliedert sich in die Vorbereitungs-, die Test- und die Auswertungsphase.

In der *Vorbereitungsphase* für den Programmtest bestehen die grundlegenden Arbeitsschritte darin, den Quellcode einzulesen, die Art der Instrumentierung zu bestimmen und die Quelldateien zu instrumentieren. Im manuellen Programmtest und in der Zusammenarbeit mit einem externen Testsystem geschieht dies über die graphische Benutzeroberfläche. Bei der Nutzung von SOTA-ATM im automatischen Testsystem kann dieses Verhalten durch Aufruf entsprechender Bibliotheksroutinen vorgenommen werden, oder SOTA-ATM über Kommandozeilenparameter gestartet werden.

In der *Testphase* erfolgen die Kompilation, der Programmstart und der Programmtest. Das Kompilieren der instrumentierten Quelldateien gehört nicht mehr in den Aufgabenbereich von SOTA, allerdings kann durch die Einbindung eines entsprechenden Ant-Buildscripts die Kompilation aus SOTA heraus initiiert werden. Liegt auch noch eine passende Batchdatei vor, so lässt sich über diese auch das zu testende Programm aus SOTA heraus starten. Werden diese beiden Dateien in SOTA eingebunden, kann der manuelle Programmtest ohne Rückgriff auf eine externe Entwicklungsumgebung vollständig aus SOTA heraus vollzogen werden. Während des Programmtests produzieren die in den Quellcode eingefügten Instrumentierungen nun eine Logdatei, welche die nötigen Informationen zur vollständigen Rekonstruktion des Programmablaufes enthält.

In der *Auswertungsphase* wird schließlich der originale Quellcode wiederhergestellt, die Logdateien werden eingelesen und die verschiedenen Überdeckungsmaße berechnet. Die Ergebnisse lassen sich dann sowohl in SOTA auf verschiedene Arten visuell auswerten, als auch in einen HTML-Report exportieren.

SOTA im Testbetrieb

Phasen	Aufgaben	Nutzungsart		
		Manueller Programmtest	Mit externem Testsystem (ATOSj)	Im automatischen Testsystem
Vorbereitungsphase	Einlesen und Parsen des Quellcodes	SOTA	SOTA	SOTA-ATM
	Konfiguration der Instrumentierung	SOTA	SOTA	SOTA-ATM / Konfigurationsdatei
	Instrumentierung des Quellcodes	SOTA	SOTA	SOTA-ATM
	Kompilieren des Quellcodes	Extern / aus SOTA per	Extern	Extern

Testphase		Skript		
	Programmstart	Extern / aus SOTA per Skript	Extern	Extern
	Programmtest	Manuell	Extern	Extern
Auswertungsphase	Restauration des originalen Quellcodes	SOTA	SOTA	SOTA-ATM
	Einlesen der Logdateien und Berechnung der Überdeckungsmaße	SOTA	SOTA	SOTA-ATM
	Visualisierung der Ergebnisse	SOTA	SOTA	Keine / SOTA (nach den Tests)
	Reporterstellung	SOTA	SOTA	SOTA-ATM

SOTA Benutzer-Dokumentation - Version 1.0

3 Installation und Programmstart

SOTA wurde als eine Standalone-Eclipse-RCP-Application entwickelt und stellt außer einer aktuellen Java Installation ab Java 6.0 keine weiteren Anforderungen an das System. Ein einfaches Aufrufen der SOTA.exe startet das Programm. Beim ersten Programmstart wird die Eclipse-Rich-Client-Platform für SOTA konfiguriert.

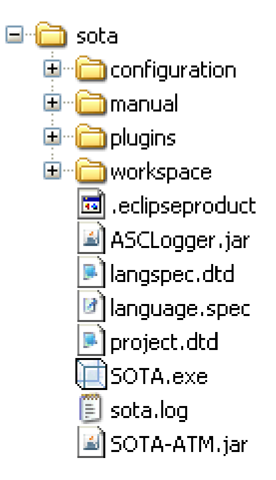


Abb.: SOTA-Verzeichnisstruktur

Abhängig von der Nutzungsart müssen weitere Systeme installiert sein, wie z.B. Eclipse oder ATOSj.

SOTA Benutzer-Dokumentation - Version 1.0

4 Benutzeroberfläche und Funktionalität

Neben der obligatorischen Menüleiste und der Toolbar für den schnellen Zugriff auf die gebräuchlichsten Aktionen besteht die Benutzeroberfläche aus verschiedenen Views, die in drei Bereichen per Tabs einsehbar sind. Im linken oberen Bereich befindet sich die View mit der Übersicht über das Projekt. Darunter sind zwei Views angeordnet, die zum einen die Testfälle und zum anderen die Instrumentationsschemata des Projektes auflisten. Den größten Teil des Fensters nimmt der rechte Bereich ein, dessen verschiedene Views umfangreiche Informationen zum Quellcode, zu den Kontrollflussgraphen, Überdeckungs- und Metrikauswertungen liefert.

Die Views sind miteinander verknüpft, so dass z.B. die Auswahl einer Datei in der ProjectView den entsprechenden Quellcode im SourceView anzeigt. Die Auswahl eines Tests in der TestView aktualisiert daraufhin die Überdeckungsmaße für das gesamte Projekt und die Darstellung der Überdeckung für die ausgewählte Datei. Für Eingaben bei größeren Aktionen wie Projekterstellungen werden Wizards oder Dialoge geöffnet.

Am unteren Fensterrand befindet sich schließlich die Statuszeile, die zum einen Informationen über den Zustand der Quelldateien liefert und zum anderen anzeigt, wie der Status des durch SOTA geparsten und daraufhin angezeigten Quellcodes ist. Der originale Zustand wird dabei als CLEAN und der

instrumentierte als DIRTY gekennzeichnet.

Abb.: Hauptfenster von SOTA

4.1 Menüs und Toolbar

4.1.1 Menü *Project*



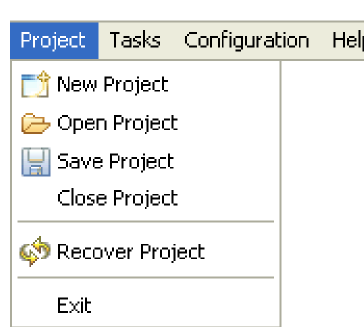


Abb.: Menü *Project*

New Project

Der Menüpunkt *New Project* öffnet den zweiseitigen Wizard für die Projekterstellung. Auf der ersten Seite muss ein Name für das Projekt angegeben werden, unter welchem SOTA dieses verwaltet, sowie das Projektverzeichnis und das Ausführungsverzeichnis. Aus dem Projektnamen generiert SOTA die Datei <projektname>.project und legt diese im SOTA-Verzeichnis ab.

Das Projektverzeichnis ist dabei das Basisverzeichnis des Testprogrammes (entspricht bei Eclipse-Projekten "../workspace/projektname"). Von hier ausgehend werden die Quellcodes importiert und in dieses Verzeichnis wird auch der Testreport geschrieben. Das Ausführungsverzeichnis ist das Verzeichnis, aus welchem das Testprogramm beim Test gestartet wird. Dies entspricht in den meisten Fällen dem Basisverzeichnis des Testprogrammes, jedoch bei Eclipse-RCP-Projekten dem Eclipse-Basisverzeichnis "..\eclipse\". In dem Ausführungsverzeichnis wird die ASCLogger.ini erstellt, die beim Test vom ASCLogger eingelesen werden muss und von hier aus werden auch nach dem Test die Testlogs eingelesen.

Die Wahl der verwendeten Programmiersprache ist obligatorisch, in der aktuellen Version aber auf Java beschränkt. Abschließend können noch eine Ant-Buildfile zum Kompilieren und eine Batchdatei zum Starten des Testprojektes eingebunden werden.

Auf der zweiten Wizardseite werden die Quelldateien des Projektes eingebunden, die im Test berücksichtigt werden. Dazu können Verzeichnisse ausgewählt werden, wodurch alle Quelldateien aus den Unterverzeichnissen importiert werden, oder aber nur einzelne Dateien.

Nach dem Beenden des Wizards wird das Projekt erstellt und die ausgewählten Quelldateien werden geladen.

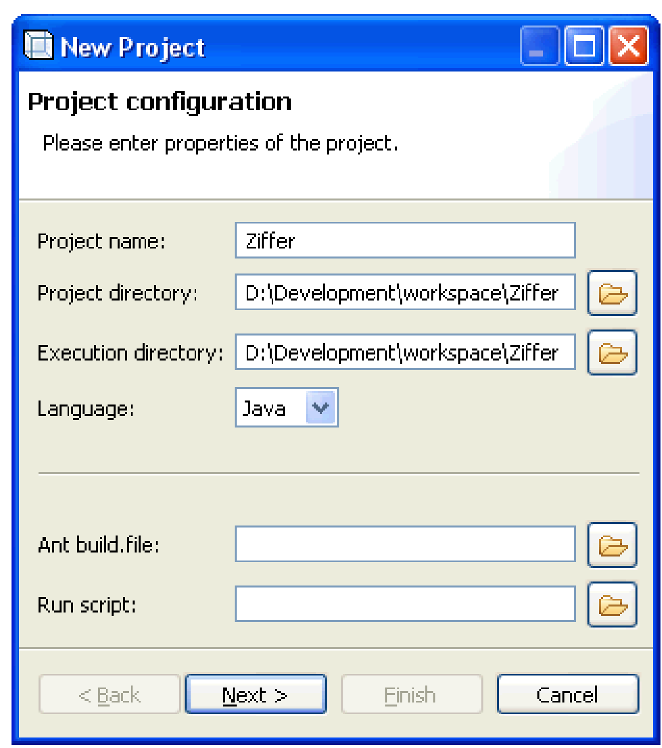


Abb.: erste Wizardseite

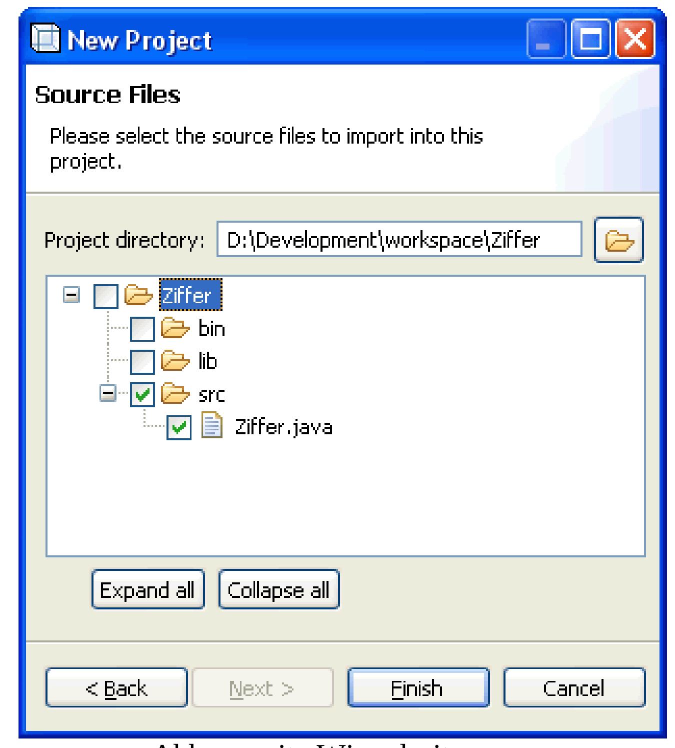


Abb.: zweite Wizardseite

Open Project



Mit dem Menüpunkt *Open Project* lässt sich ein zuvor erstelltes Projekt öffnen. Dazu wird ein Standarddialog zum Laden einer Datei im Basisverzeichnis von SOTA geöffnet, in welchem man das entsprechende Projekt auswählen kann.
Nach der Bestätigung durch *OK* wird das Projekt in SOTA geöffnet und die dem Projekt zugehörigen Quelldateien werden geladen.

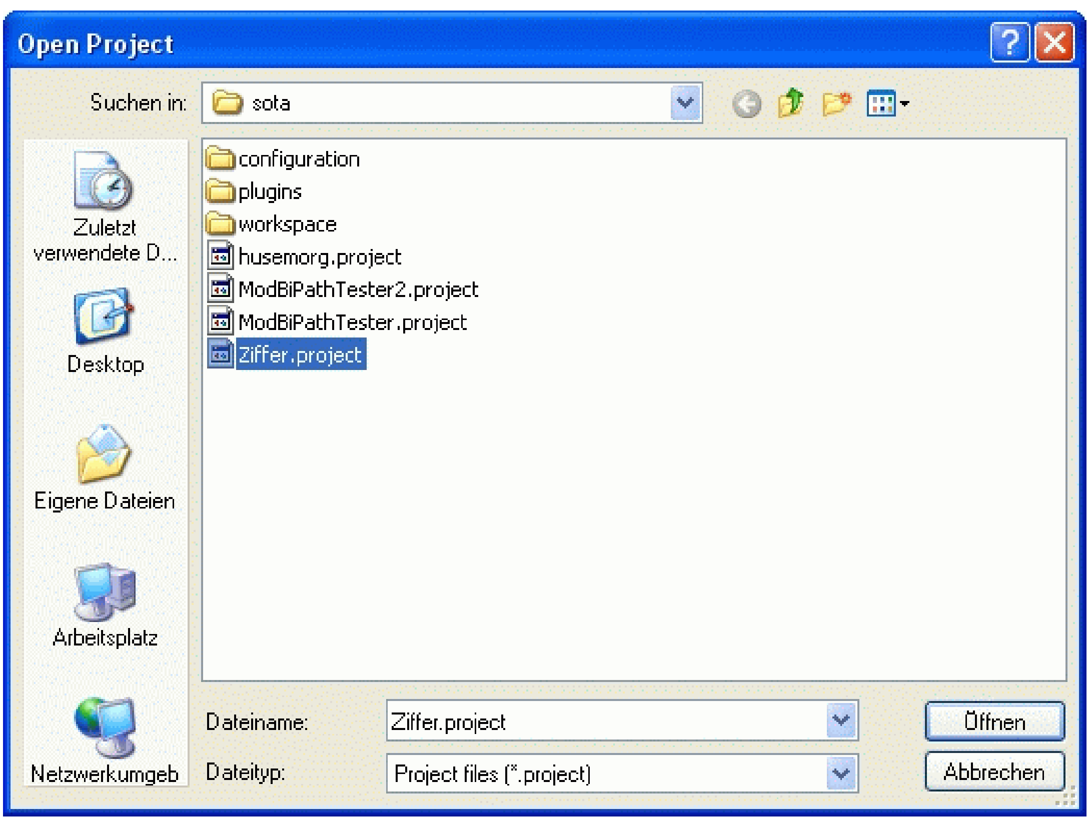


Abb.: Dialog *Open Project*

Save Project



Das Speichern des Projektes über den Menüpunkt *Save Project* führt beim erstmaligen Ausführen zur Erstellung der Projektdatei im Basisverzeichnis von SOTA, dessen Name identisch mit dem eingegebenen Projektnamen ist und auf "project" endet. Diese Datei enthält die Projektdaten und auch alle erstellten InstrumentationsSchemata. Existiert diese Datei schon, wird sie mit den aktuellen Daten überschrieben.

Close Project

Über den Menüpunkt *Close Project* lässt sich das aktuelle Projekt schließen. Sota befindet sich daraufhin wieder im Startzustand.

Recover Project

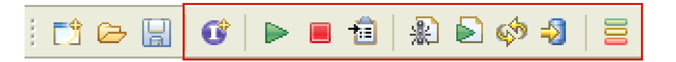


Der Menüpunkt *Recover Project* hat die Aufgabe, korrupte Projekte wiederherzustellen. Dazu wird analog zum Menüpunkt *Open Project* ein Dialog geöffnet, womit man ein Projekt auswählen kann. Für dieses Projekt werden alle Quelldateien wieder in ihren Originalzustand zurückversetzt und daraufhin das Projekt geöffnet.

Exit

Über den Menüpunkt *Exit* wird SOTA beendet.

4.1.2 Menü *Tasks*



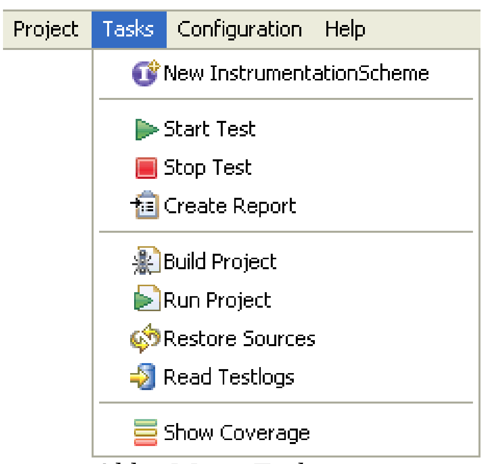


Abb.: Menü *Tasks*

New InstrumentationScheme



Der Menüpunkt *New InstrumentationScheme* öffnet einen Dialog, der das Erstellen eines neuen Instrumentationsschemas (kurz: IScheme) für das aktuelle Projekt ermöglicht. Ein IScheme entspricht einer Instrumentierungsvorgabe für alle Quelldateien des Projekts, die jeweils verschiedene Grade annehmen kann. Der Sinn dieser Konfigurationsmöglichkeit für die Instrumentierung besteht darin, dem Nutzer zu erlauben, den abhängig vom Testfall mitunter sehr hoch ausfallenden Speicherbedarf der Logdateien sinnvoll zu begrenzen.

Für die Erstellung eines ISchemes muss im Dialog ein Name angegeben werden, unter dem dieses im Projekt verwaltet wird. Die Angabe einer Beschreibung ist optional. Die sich darunter befindliche Projektansicht zeigt hierarchisch alle Strukturen des aktuellen Projektes in Baumform, beginnend von Dateien über Klassen zu den einzelnen Funktionen. Jeder der Strukturen kann ein Level der Instrumentierung von 0 bis 3 zugewiesen werden, indem man nach der Auswahl der Struktur den entsprechenden Button für das Level drückt, wodurch diese Belegung auch farblich gekennzeichnet wird. Dabei gibt das Zuweisen eines Instrumentierungslevels für eine Struktur dieses auch an alle untergeordneten Strukturen weiter. Die einzelnen Level haben folgende Auswirkung:

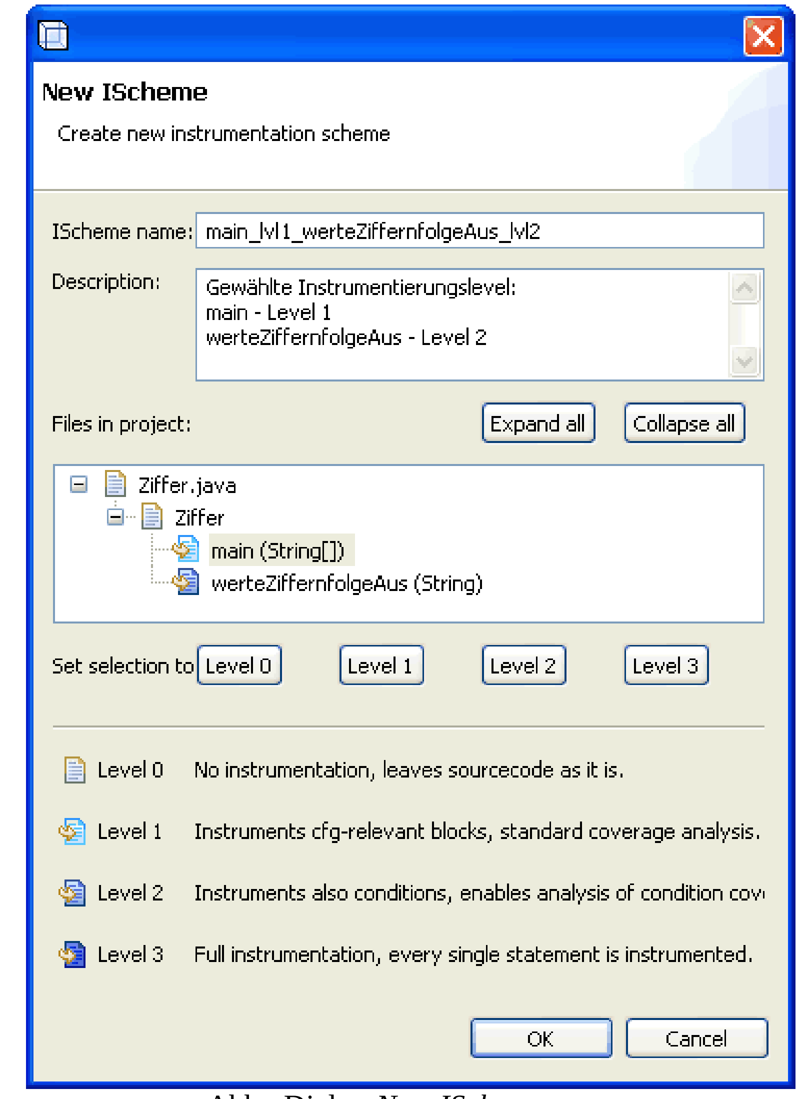


Abb.: Dialog *New IScheme*

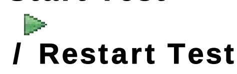
- Level 0 - der Quellcode wird nicht instrumentiert,
- Level 1 - die nötigen Anweisungen werden instrumentiert, so dass die kontrollflussrelevanten Überdeckungsmaße ermittelt werden können (FEEC, C0, C1, MBI, BI),
- 
  Level 2 - zusätzlich zu Level 1 werden auch alle Bedingungen instrumentiert, so dass die Bedingungsüberdeckungsmaße berechnet werden können (C2, MMCC, MCDC, C3),

- 
  Level 3 - der Quellcode wird vollständig instrumentiert, d.h. jede Anweisung und alle Bedingungen werden ausgewertet.

Das empfohlene Instrumentationslevel für den Programmtest ist Level 2, welches die Ermittlung aller Überdeckungsmaße erlaubt. Im Bedarfsfall kann man dieses auf ein niedrigeres Level reduzieren, um Kapazitäten zu sparen. Die Verwendung von Level 3 ist dort sinnvoll, wo man bei nicht korrekt beendenden Programmen bzw. Funktionen die genaue Abbruchstelle bzw. den Ort des Ausnahmefalles ermitteln möchte.

Nach dem Bestätigen des Dialoges werden das Instrumentationsschema mitsamt der ausgewählten Einstellung in der projektspezifischen .ischeme-Datei gesichert und steht ab sofort für den Programmtest zur Verfügung.

Start Test / Restart Test



Der Programmtest wird in SOTA durch den Menüpunkt *Start Test* eingeleitet, welcher eine Dialogbox zum Konfigurieren des Tests öffnet. Der eingegebene Name für den Test dient gleichzeitig als Name für das dann zu erstellene Testlog, d.h. bei der Wahl des Namens muss auf die Beschränkungen von Dateinamen für das jeweilige Betriebssystem Rücksicht genommen werden. Eine Beschreibung des Tests kann zusätzlich hinzugefügt werden, ist jedoch nicht zwingend.

Als nächstes ist es notwendig, die Instrumentierung des Projektes zu konfigurieren. Dazu stehen neben den selbsterstellten ISchemes die drei elementaren ISchemes zur Verfügung, die das gesamte Projekt nach Level 1, Level 2 sowie Level 3 instrumentieren. Die hierarchische Projektübersicht zeigt die konkrete Instrumentierung der einzelnen Projektstrukturen für das gewählte IScheme. Mit den beiden Buttons *Expand All* sowie *Collapse All* lässt sich die Baumstruktur komplett auf- bzw. zuklappen.

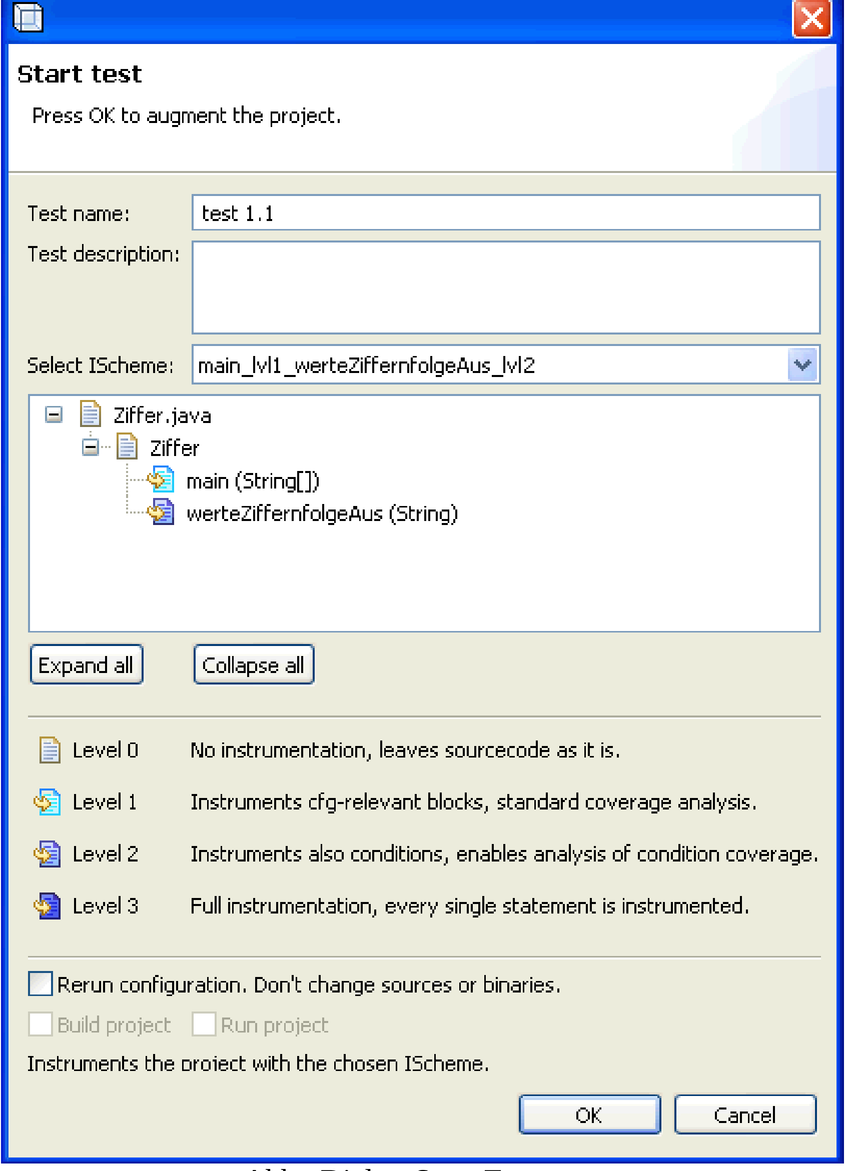


Abb.: Dialog *Start Test*

Sobald ein Name für den Test eingegeben und ein IScheme ausgewählt wurde, kann der Programmtest durch bestätigen des Dialoges gestartet werden. Die Informationen für die Loggingkomponente wird in

die Datei "ASCLogger.ini" in den Ausführungspfad des Projektes geschrieben, es folgt die Sicherung der originalen Quelldatein durch eine Änderung der Dateiendungen in ".backup" und die Instrumentierung des Quellcodes. Daraufhin wird das Projekt erneut geparst und der instrumentierte Quellcode kann in der SourceView betrachtet werden. Der Quelltext liegt nun in instrumentierter Fassung vor und kann kompiliert und z.B. mit einem externen Testsystem systematisch getestet werden.

Drei weitere Optionen sind am unteren Rand durch Checkboxen auswählbar. *Rerun configuration* führt dazu, dass lediglich der Name des Tests und die Beschreibung in der ASCLogger.ini geändert wird, aber keine Änderung am Quelltext vollzogen wird. Dies ermöglicht einen erneuten Test mit der gleichen Konfiguration, ohne dass noch einmal instrumentiert und kompiliert werden muss. *Build project* veranlasst SOTA nach dem Instrumentieren das zu testende Projekt zu kompilieren. Dazu ist es zum einen notwendig, in den Präferenzen unter dem Punkt *General* eine Ant-Version über die Angabe des Pfades der "ant.bat" einzubinden und zum andern, zum Projekt eine entsprechende xml-Datei hinzufügen, die das Kompilieren mittels Ant ermöglicht. Diese Option ist nur auswählbar, wenn ein entsprechendes Skript vorliegt und nicht die Option *Rerun configuration* ausgewählt wurde. Schließlich erlaubt die Option *Run project* das Projekt nach dem Kompilieren auch zu starten, sofern ein entsprechendes Startskript für das Projekt angegeben wurde. Das Starten ist nur möglich, wenn eine der beiden anderen Optionen ausgewählt wurde.

Wurde ein Test gestartet oder findet SOTA beim Parsen der Quellen diese schon instrumentiert vor, dann ändert sich die Option



Start Test zu



Restart Test. Es ist nicht möglich, die Quellen neu zu instrumentieren, ohne dass der aktuelle Test beendet und die Originaldateien wiederhergestell worden sind. Stattdessen besteht die Möglichkeit, das vorliegende instrumentierte Testprogramm erneut unter einem neuen Namen für den Test zu starten. Dabei wird lediglich die Initialisierungsdatei für die Loggingkomponente mit dem neuen Namen für den Test aktualisiert, jedoch werden keine Quelldateien oder Binaries verändert. Dieser Neustart entspricht der *rerun*-Option im normalen *Start-Test*-Dialog.

Stop Test



Der Menüpunkt *Stop Test* beendet den aktuellen Testlauf. Die originalen Quelldateien werden daraufhin wiederhergestellt sowie neu eingelesen. Liegt das entsprechende Ant-Buildfile vor, werden die originalen Quellen auch wieder neu kompiliert.

Zum Abschluss wird ein Dialog geöffnet, der es erlaubt Logdateien aus dem Ausführungsverzeichnis auszuwählen, woraufhin diese eingelesen, ausgewertet und daraufhin in der View *TestLogs* aufgelistet werden. Im Anschluss daran lassen sich die Überdeckungsmaße für einzelne Testlogs oder Kombinationen von mehreren berechnen und anzeigen.

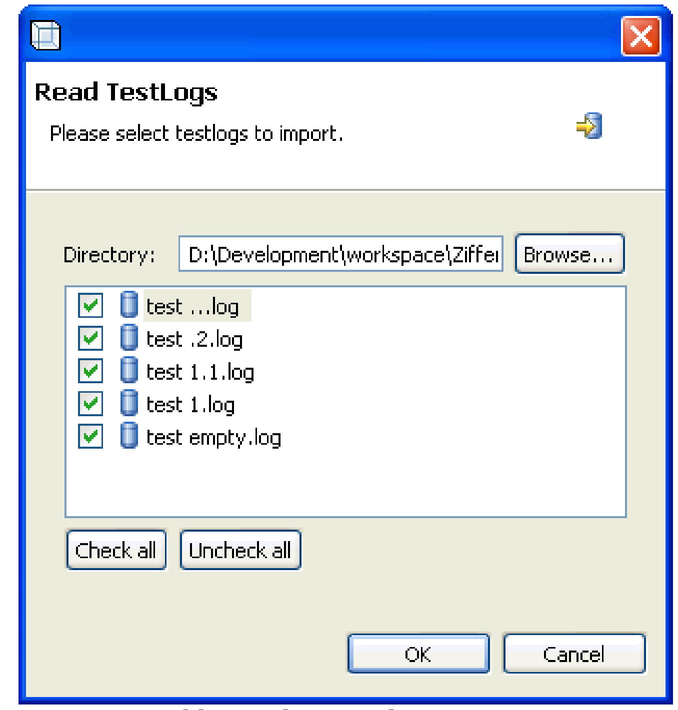


Abb.: Dialog *Read Logs*

Create Report



Über den Menüpunkt *Create Report* lässt sich ein

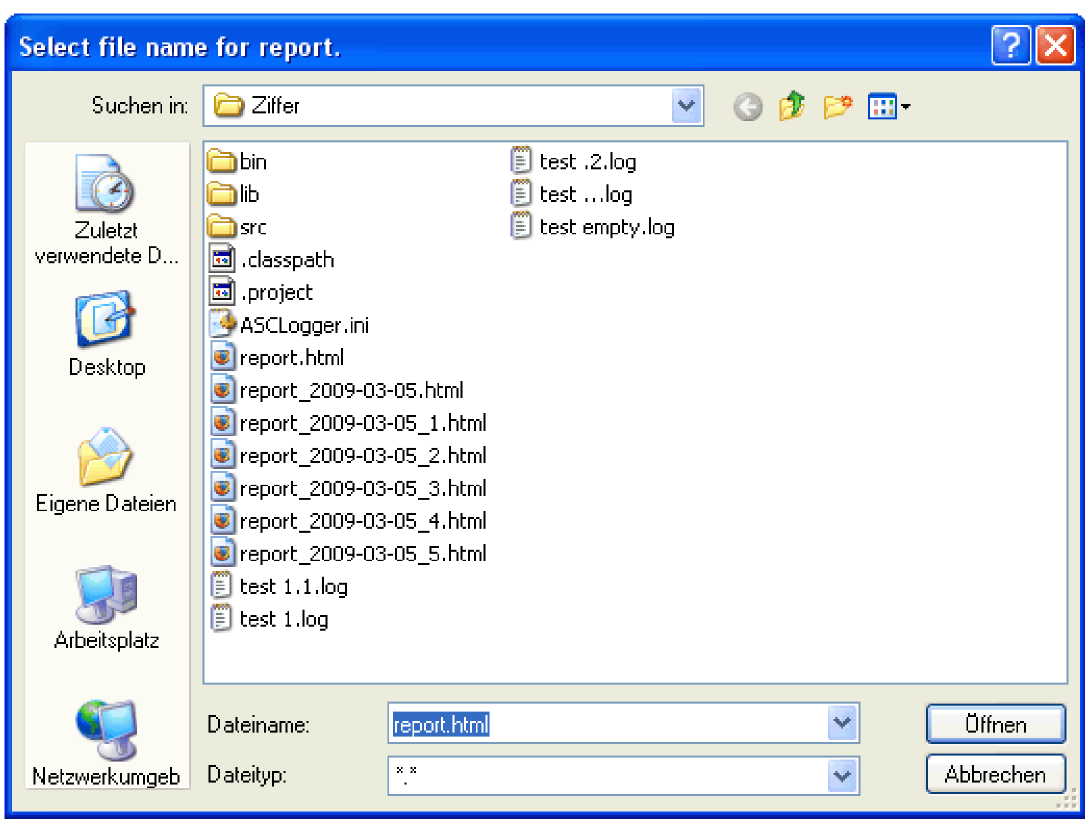


Abb.: Dialog *Create Report*

Überdeckungsreport anhand der eingelesenen Testlogs für das aktuelle Projekt erstellen. Wurde in den Präferenzen eingestellt, dass bei der Reporterstellung nach einem Dateinamen gefragt werden soll, öffnet sich ein Dialog zur Bestimmung der Reportdatei. Ansonsten wird der Report im Basisverzeichnis des Projekts unter dem Namen "report.html" gespeichert. Falls in den Präferenzen bestimmt wurde, dass diese Datei nicht überschrieben werden soll, so wird für jeden Report eine neue Datei nach folgendem Format generiert: "report_<date>_<index>.html". Diese und weitere Einstellung zur Reporterstellung lassen sich im Präferenzenmenü unter *Report* vornehmen

Build Project



Der Menüpunkt *Build Project* erlaubt unabhängig vom Teststatus das Kompilieren des Projektes aufgrund der aktuellen Quelldateien aus SOTA heraus. Dieser Menüpunkt ist nur dann aktiviert, wenn ein xml-Buildscript für das aktuelle Projekt angegeben und eine Ant-Version in den Präferenzen eingebunden wurde.

Run Project



Der Menüpunkt *Run Project* ist wählbar, wenn ein Startskript für das aktuelle Projekt angegeben wurde und führt dazu, dass dieses ausgeführt wird.

Restore Sources



Um das Projekt wieder in den Originalzustand zu überführen, kann auch der Menüpunkt *Restore Sources* gewählt werden. Im Gegensatz zum Menüpunkt *Stop Test* werden hier ausschließlich die Quelldateien wiederhergestellt und eingelesen, es wird weder neu kompiliert (so ein Buildfile vorliegt) noch wird der Dialog für das Einlesen der Testlogs geöffnet.

Read Logs



Mit dem Menüpunkt *Read Logs* kann man manuell die Logdateien unabhängig vom Teststatus des Projektes einlesen. Es wird dazu derselbe Dialog wie im Menüpunkt *Stop Test* geöffnet, der das Einlesen der Logdateien ermöglicht. Nach der Auswahl der Logdateien aus dem Ausführungsverzeichnis des Projektes, folgt wiederum das Einlesen und Aufbereiten der Testergebnisse.

Show Coverage



Die Aktivierung des Menüpunkts *Show Coverage* führt dazu, dass in der Quellcode- und der Überdeckungsansicht die Überdeckungsangaben farbig dargestellt werden.

4.1.3 Menü *Configuration*

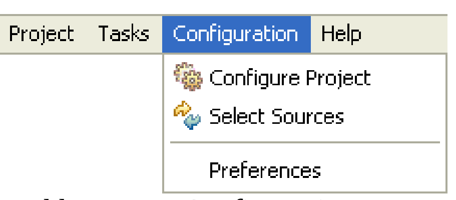


Abb.: Menü *Configuration*

Configure Project



Projektspezifische Einstellungen können über den Menüpunkt *Configure Project* vorgenommen werden. Im Einzelnen sind dies die beiden für SOTA relevanten Pfade des Testprojektes, nämlich das Basisverzeichnis sowie das Ausführungsverzeichnis, und darunter die eventuell vorliegenden Skripte, die zum Zweck der Kompilation und des Startens des Projektes in SOTA an dieser Stelle eingebunden werden können. Diese Werte werden nach Bestätigung durch *Ok* die bei der Projekterstellung eingegebenen Werte überschreiben.

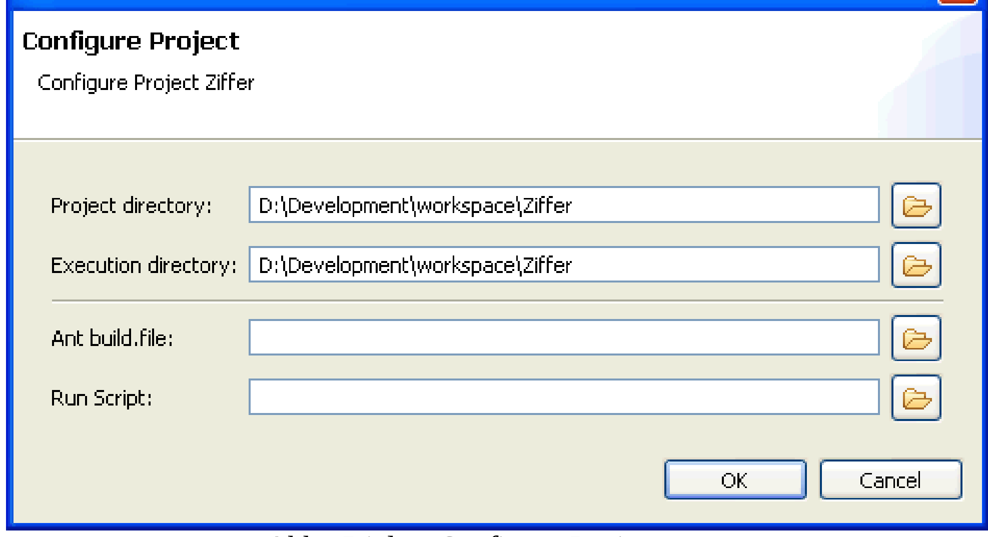


Abb.: Dialog *Configure Project*

Select Sources



Möchte man die vom Projekt umfassten Quelldateien erweitern oder ändern, ist dies über den Menüpunkt *Update Sources* möglich. Der sich daraufhin öffnende Dialog ist identisch zur zweiten Wizardseite der Projekterstellung. Hier lassen sich dazu analog die Quelldateien für das Projekt auswählen. Mit der Bestätigung der Auswahl werden die aktuellen Quelldateien durch die neuausgewählten ersetzt.

Preferences

Über den Menüpunkt *Preferences* lassen sich allgemeine Einstellungen zu SOTA durchführen, die für alle Projekte gelten und auch beim Verlassen des Programmes gespeichert werden. Die genaue Erläuterung der umfangreichen Einstellungsmöglichkeiten erfolgt unter dem Punkt 4.3 Präferenzen.

4.1.4 Menü *Help*

Manual



Durch Auswahl des Menüpunktes *Manual* lässt sich die

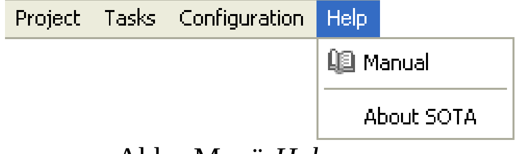


Abb.: Menü *Help*

vorliegende Benutzerdokumentation aus SOTA heraus im Standardsystembrowser öffnen.

About

Der Menüpunkt *About* öffnet den Info-Dialog mit den Informationen zur benutzten SOTA-Version und den Plugin-Status der Applikation.

4.2 Views

4.2.1 View *Project*

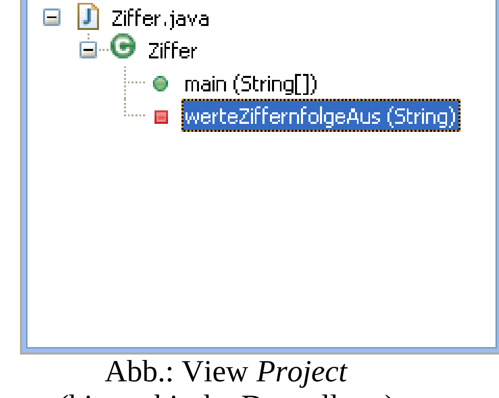


Abb.: View *Project* (hierarchische Darstellung)

In der View *Project* werden die Quellcodedateien des Testprogrammes und ihre untergeordneten Strukturen, wie Klassen und Methoden als Baum aufgelistet. Die obersten Knoten stehen für die Quellcodedateien, die man bei der Projekterstellung importiert hat. Als Kinder werden dann hierarchisch geordnet die Toplevel-Klassen und -Methoden und ihre jeweiligen inneren Klassen und Methoden bis zu einer beliebigen Schachtelungstiefe aufgeführt. Mittels der beiden Buttons  und kann der Baum komplett expandiert bzw. kollabiert werden. Der Button  lässt die Liste abwechselnd aufsteigend und absteigend sortieren. Die Auswahl eines Element der Baumstruktur aktualisiert automatisch die Views *Source* und *CFG*.

Die einzelnen in dieser View aufgelisteten Strukturen sind dabei:



- Dateien
- Klassen
- innere Klassen
- public-Funktionen
- protected-Funktionen
- private-Funktionen

Wenn beim Parsen der Datei Instrumentationsanweisungen festgestellt werden, wie z.B. während des Testlaufes auf instrumentierten Quellen, so wird auf dem Icon zusätzlich noch ein rotes Ausrufezeichen angezeigt (Bsp.:).

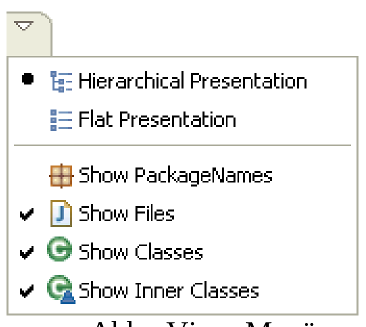


Abb.: View-Menü

Über das rechts oben in der View auswählbare Menü lässt sich die Darstellung der Projektstrukturen konfigurieren. Hier lassen sich die zwei Darstellungsoptionen



Hierarchical Presentation und

Flat Presentation auswählen. Die hierarchische Darstellung entspricht der oben beschrieben Darstellung als Baumstruktur, welche die Standardeinstellung ist. Bei der flachen Darstellung liegen alle Struktureinheiten auf der Wurzelebene des Baumes, d.h. Datei, Klassen und Funktionen werden gleichrangig aufgelistet.

Abb.: View *Project* (flache Darstellung ohne Dateien)

Mit den weiteren Optionen lässt sich die Darstellung wie folgt einstellen:

- *Show PackageNames* - ergänzt den Namen der Struktur durch den Paketnamen (default: aus)
- *Show Files* - führt in der View auch Dateien auf (default: an)
- *Show Classes* - führt in der View Klassen auf. Hat nur bei der flachen Darstellung einen Effekt, bei der hierarchischen Darstellung werden immer Klassen dargestellt. (default: an)
- *Show Inner Classes* - führt in der View innere Klassen auf (default: an).

4.2.2 View *Testlogs*

Wurden Logdateien in das Projekt eingelesen, so werden diese in der View *TestLogs* aufgelistet. Falls die Logdatei schon vorhanden ist, werden die Logdaten an deren Ende angefügt, so dass ein Testlog eine Vielzahl von Testfalldaten enthalten kann. Als Standardeinstellung werden die einzelnen Testfälle vor dem Nutzer verborgen und lediglich die Testlogs aufgeführt. Um auch detailliert den Inhalt der Testlogs anzeigen zu lassen, muss man in den Toolbar der View den Button



Lock Testlogs entriegeln, woraufhin zu jedem Testlog unter diesem alle in ihm enthaltenen Testfälle samt IScheme aufgelistet werden.

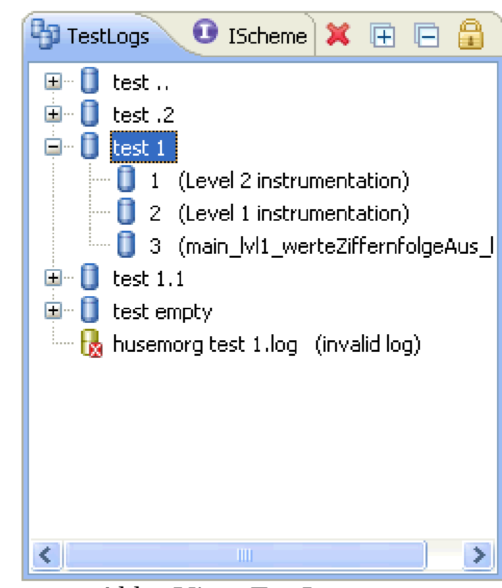


Abb.: View *TestLogs* (unlocked)

Nicht jede Logdatei ist von SOTA erstellt worden und nicht alle Testfalldaten gehört zu dem aktuellen Projekt. Die View *TestLogs* verwendet daher folgende Icons, um die Kompatibilität darzustellen:

-

- vollständig kompatible Testlog bzw. Testfalldaten
- teilweise kompatibles Testlog, enthält auch invalide Testfalldaten
- invalide Logdatei oder inkompatible Testfalldaten

Wählt man valide Testlogs bzw. Testfälle aus, werden automatisch die Überdeckungsinformation in den Views *Source*, *CFG* und *Coverage* entsprechend aktualisiert. Zum Ändern der kompletten Auswahl können die Buttons



und

verwendet werden. Eine Mehrfachauswahl von einzelnen Testlogs kann durch das Gedrückthalten der Tasten <Shift> bzw. <Strg> mit anschließender Auswahl der gewünschten Testlogs erreicht werden.

Ausgewählte Testlogs können über den Button



Delete TestLogs gelöscht werden. Sie werden dann nicht nur aus dem Projekt sondern aus dem System entfernt. Diese Option steht nicht für Testfalldaten, also einzelne Teile eines Testlogs zur Verfügung.

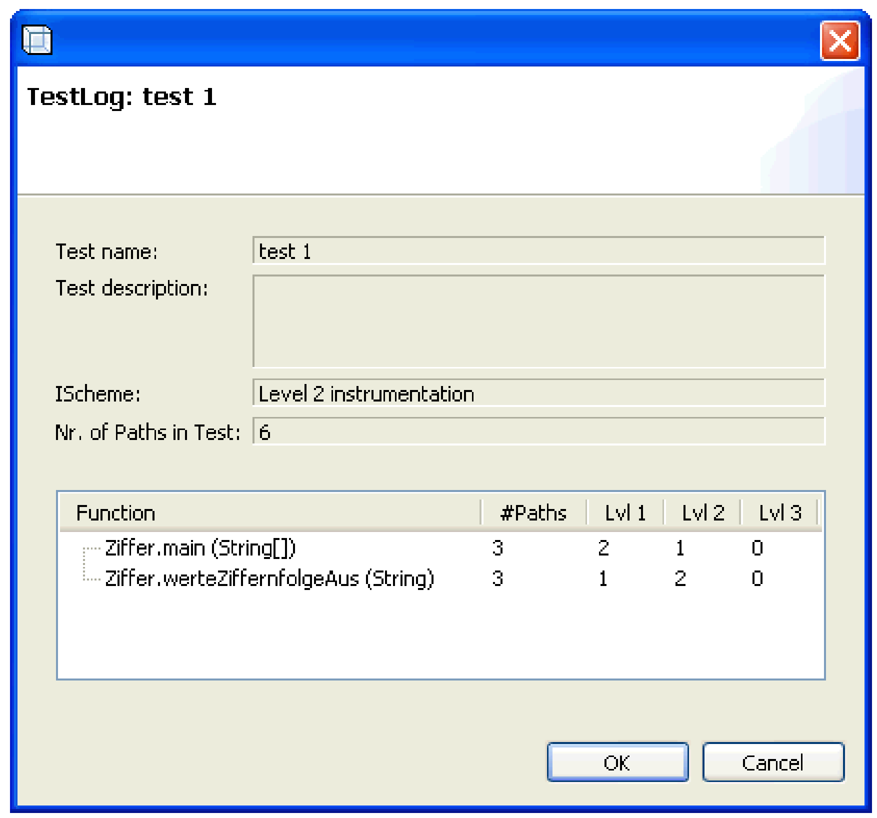


Abb.: Dialog *TestLogs*

Durch einen Doppelklick auf einen der aufgelisteten Testlogs bzw. Testfalldatensatz lässt sich ein Dialog öffnen, welcher detaillierte Informationen zu den Testdaten enthält. Hier werden zuerst der Name und die Beschreibung des Tests sowie der Name des Instrumentierungsschemas aufgeführt. Als nächstes erfolgt zum einen die Angabe der Anzahl an Pfaden (d.h. Durchläufen einer Funktion), die der Testfall insgesamt enthält, sowie eine Auflistung aller durch diesen Test berührten Funktionen. Zu diesen Funktionen wird außerdem aufgelistet, wieviele Pfade zu ihr gehören. Die Liste lässt sich durch das Anklicken der Spaltenköpfe jeweils auf- und absteigend alphabetisch nach den Funktionsnamen und der Anzahl der Pfade sortieren.

4.2.3 View *IScheme*

In der View *IScheme* befinden sich lediglich die zum Projekt gehörigen Instrumentationsschemata, kurz Ischemes genannt. Zu jedem Projekt werden automatisch drei Ischemes erstellt, die jeweils das gesamte Projekt nach Level 1, 2 bzw. 3 instrumentieren. Zusätzliche ISchemes können über den Menüpunkt *New IScheme* erstellt werden

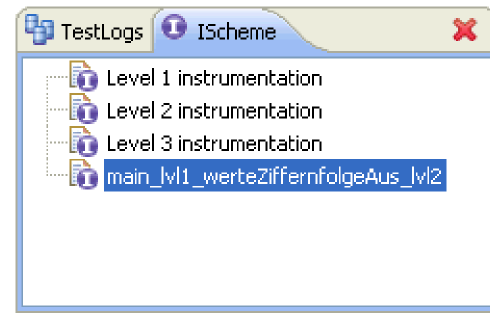


Abb.: View *ISchemes*

und erscheinen danach in dieser View.

Durch Doppelklicken auf ein IScheme öffnet sich sich ein Dialog, der analog zum Menüpunkt *New IScheme* die gespeicherten Instrumentierungseinstellungen des ISchemes auflistet und die Änderung sämtlicher Informationen erlaubt. Über den Button



Delete IScheme kann ein IScheme auch aus dem Projekt gelöscht werden.

Die werkseigenen ISchemes über Level 1,2 und 3 sind von diesen Änderungen ausgenommen und können auch nicht gelöscht werden.

4.2.4 View *Source*

In der View *Source* wird der Quellcode der ausgewählten Datei aus der View *Project* angezeigt. Werden Klassen oder Methoden ausgewählt, werden nur die entsprechenden Quellcodezeilen angezeigt, die zu dieser Einheit gehören.

Ist die Überdeckungsanzeige



***Show Coverage* in der Toolbar aktiviert, wird die Überdeckung des Quelltextes durch die ausgewählten Testlogs der View *TestLogs* farbig dargestellt. Grüne Quellcodezeilen wurden durch die Tests abgedeckt, rote wurden im Test nicht berührt. Liegen mehrere Anweisungen in einer Zeile, wird die Zeile grün markiert, wenn auch nur eine einzige der Anweisungen überdeckt wurde. Die entsprechenden Farben für die Zeilenüberdeckung sowie das Syntaxhighlighting können im Menüpunkt *Preferences* eingestellt werden.**

Source | CFG | Coverage | Metrics

```
public class Ziffer {

    public static void main(String[] args) {
        System.out.println(werteZiffernfolgeAus("."));
    }

    private static double werteZiffernfolgeAus(String inZiffernString) {

        double wert = 0.0;
        double genauigkeit = 1.0;
        String woBinIch = "VorKomma";
        boolean fehlerfrei = true;
        int position = 1;
        while(position <= inZiffernString.length() &fehlerfrei) {
            String zchn = inZiffernString.substring(position-1, position);
            if(zchn.matches("[0-9]")) {
                if(woBinIch.equals("NachKomma"))
                    genauigkeit = genauigkeit / 10.0;
                wert = 10.0*wert + Double.parseDouble(zchn);
            }
            else if(zchn.equals(".")  & woBinIch.equals("VorKomma"))
                woBinIch  = "NachKomma";
            else
                fehlerfrei = false;

            position ++;
        }
        if(!fehlerfrei | inZiffernString.length()==0|((woBinIch.equals("NachKomma")&inZiffe
            return -1.0;
        else
            return wert*genauigkeit;
    }
```

Abb.: View *Source*

4.2.5 View *CFG*

Während die View *Source* nur die Zeilenüberdeckung anzeigen kann und primär für den Überblick über die Quellcodeüberdeckung gedacht ist, hat die View *CFG* (Control Flow Graph) die Aufgabe, detaillierte Informationen zur Überdeckung der Quellcodestrukturen zu vermitteln. Wird in der View *Project* eine Funktion ausgewählt, so zeigt der obere Teil der View *CFG* den entsprechenden Kontrollflussgraphen an. Zu jeder Funktion wird mindestens ein Knoten für den Funktionseingang und einrr für den Funktionsausgang, mit dem alle die Funktion verlassenden Kanten verbunden sind, dargestellt. Die einzelnen verzweigenden Strukturen werden durch Knoten mit folgenden Labeln im Kontrollflussgraphen abgebildet:

- bedingte Anweisung (Kurzform): "if", "true", "if-end"
- bedingte Anweisung (Langform): "if", "true", "false", "if-end"
- abweisende Iteration/while- und for-Schleife: "iteration", "iter-body", "iter-end"
- nichtabweisende Iteration/do-while-Schleife: "do", "iter-body", "iteration", "iter-end"
- Auswahl/switch-Anweisung: "switch", "case", "default", "switch-end"
- Ausnahmen/try-Blöcke: "try", "try-block", "catch", "finally", "try-end"
- Sprunganweisungen: "break", "continue', "return", "throw"

Für die Zuordnung der Knoten zu den entsprechendne Quelllcodeteilen genügt das Anklicken eines Knotens, woraufhin im unteren Teil der View *CFG* der Quellcode auf den entsprechende Quellcodeabschnitt fokussiert und die korrespondierende Zeile gelb hinterlegt wird. Ist in der Viewtoolbar die Option

Pin SourceView ausgewählt, wird nur bei der ersten Auswahl eines Knotens die entsprechende Quellcodezeile fokussiert, danach bleibt die Quellcodeanzeige "gepinnt" und scrollt nicht mehr

automatisch. Die Auswahl der Option
Show Number of Paths zeigt links neben jeder Kante des Kontrollflussgraphen die Anzahl der Kantendurchläufe an, die in den aktuellen Tests stattgefunden haben.

Damit auch große Graphen übersichtlich eingesehen werden können, lässt sich die Darstellung des Kontrollflussgraphen über den Button
Zoom Out verkleinern und dann wieder über
Zoom In vergrößern. SOTA bietet sieben Zoomstufen an, wobei noch auf den ersten drei Zoomstufe die Bedingungsüberdeckungsanzeige möglich ist und auf der vierten auch noch die Knotenbeschriftung angezeigt wird. Bei den kleinsten drei Zoomstufen werden die Knoten nur noch als unbeschriftete Quadrate dargestellt. Genauere Information sind über den Tooltip oder den Knoten-Informationsdialog erhältlich (siehe unten).

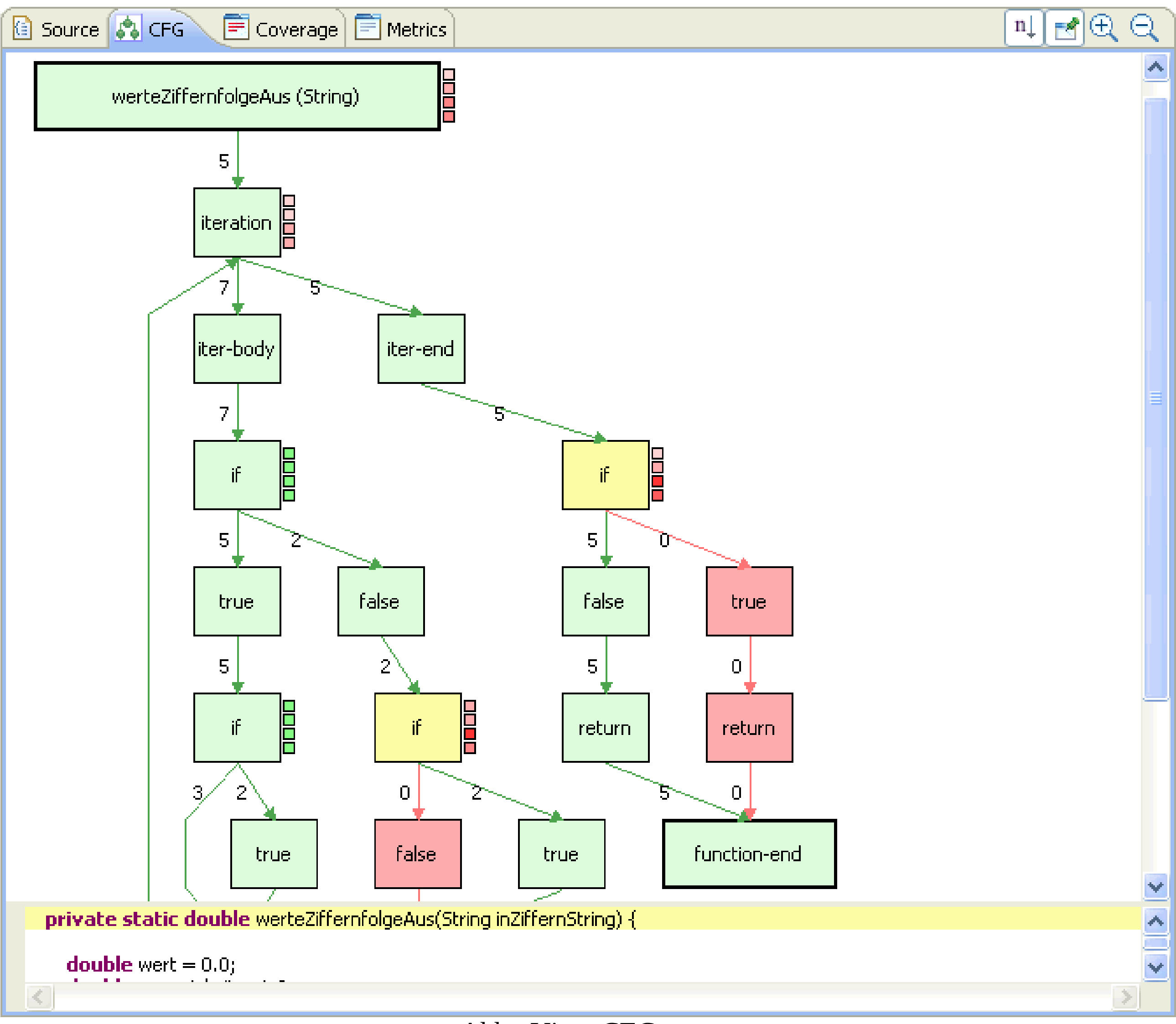


Abb.: View *CFG*

Werden in der View *TestLogs* Testlogs oder Testfalldaten ausgewählt, so ändert sich die Farbe der Knoten und Kanten entsprechend der Überdeckung durch die ausgewählte Testfallmenge. Die Überdeckungsdaten werden dabei automatisch aktualisiert. Überdeckte Knoten und Kanten werden grün dargestellt, nichtüberdeckte rot. Zur besseren Erkennung von überdeckten Knoten, die mehrere Ausgänge haben, von denen aber nicht alle überdeckt wurden – wo also die Ursache der fehlenden Überdeckung von Codeabschnitten zu suchen wäre – werden diese gelb gefärbt. Dies lässt sich aber in den Präferenzen genauso wie alle anderen Farben und auch der Linienstil sowie -dicke konfigurieren.

Zusätzliche Informationen erhält man, wenn man den Mauszeiger kurz über einem Knoten stehen lässt. Der sich öffnende Tooltip gibt zu jedem

iteration
nrHits: 5
nrSkips: 0
nrSingleLoops: 3
nrMultipleLoops: 2
nrLoops: 7
ID: 22 Line: 15

Abb.: Tooltip (Iteration-Knoten)

Knoten seinen Typ und die Anzahl der Berührungen ("nrHits") durch die Testfallmenge an, sowie seine projektinterne ID und die Zeilennummer, in welcher er zu finden ist. Verzweigende Knoten halten zudem Informationen, wie oft welche Verzweigung genommen wurde. Für If-Knoten wird daher je ein Wert für die true- und false-Verzweigung aufgeführt, Switch-Knoten halten eine Übersicht über die gewählten Selektionen und eine Auflistung, wie oft jeder Case angesprungen wurde. Im Case-Knoten wird dieser Wert, der sich von der Anzahl der Berührungen unterscheiden kann, als "nrSelects" nochmal aufgeführt. Der Tooltip für Iterationsknoten enthält neben der Anzahl der Berührungen auch noch die Angaben, wie häufig der Schleifenkörper übersprungen wurde ("nrSkips"), wie oft er genau ein einziges mal ausgeführt wurde ("nrSingleLoops"), wie oft zweimal und mehr ("nrMultipleLoops") sowie die Angabe, wie häufig der Schleifenkörper insgesamt durchlaufen wurde ("nrLoops"). Schließlich wird noch bei den try-Knoten, die die Ausnahmebehandlung einleiten, angegeben, wie oft der try-Block ohne Ausnahme abgearbeitet werden konnte.

Informationen zur Bedingungsüberdeckung kann man über zwei Wege erhalten. Zum einen befinden sich an den Knoten, die eine nicht-triviale Bedingung enthalten, an der rechten Seite vier kleine Kästchen, die für die verschiedenen

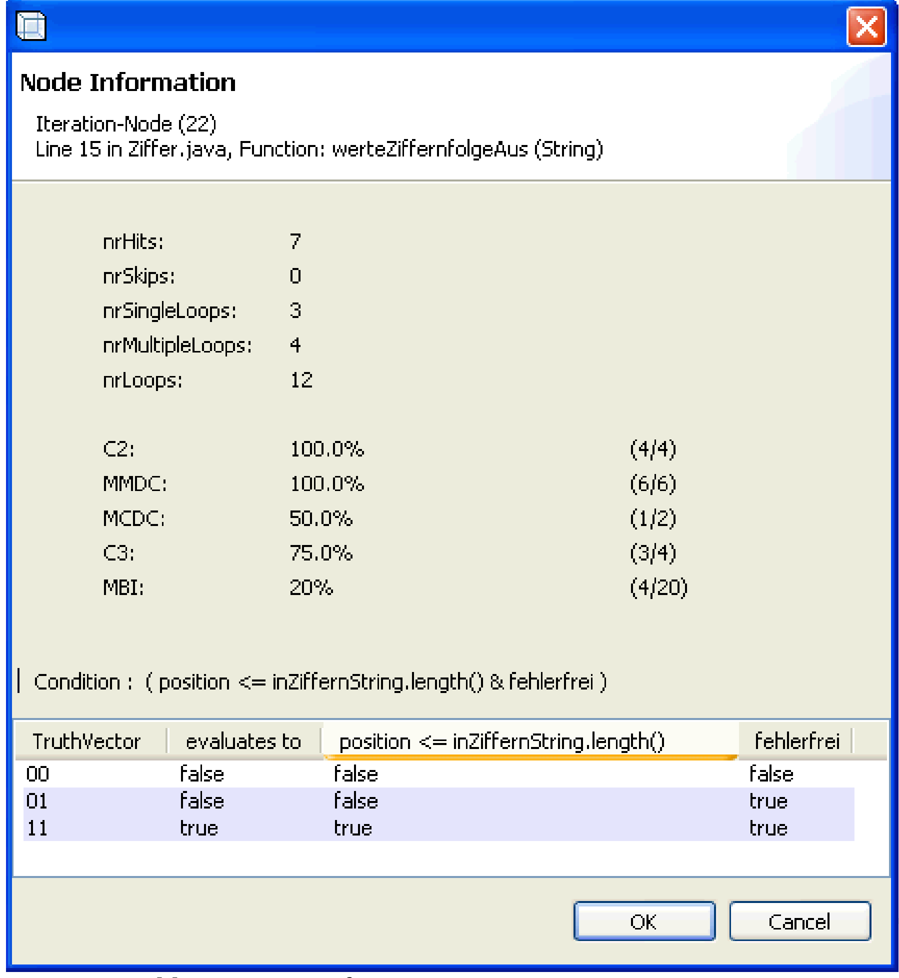


Abb.: Knoten-Information mit MCDC-Paar

Bedingungsüberdeckungsgrade stehen. Von oben nach unten sind dies die einfache, die minimal Mehrfach-, die MC/DC- sowie die Mehrfach-Bedingungsüberdeckung. Die Färbung zeigt den Grad der Überdeckung von grün gleich 100% bis dunkelrot gleich 0% an. Die Schwellwerte und Farben können in den Präferenzen festgelegt werden, wo auch ihre Darstellung abgeschaltet sowie zusätzlich noch zwei weitere Sätze an überdeckungsanzeigenden Kästchen eingeblendet werden können.

Für genauere Informationen über die Überdeckung einzelner Knoten mit oder ohne Bedingung kann durch Doppelklicken auf diesen eine Dialogbox geöffnet werden. Als erstes werden hier sämtliche Informationen des Tooltips und zusätzlich dazu noch alle für diesen Knoten relevanten Überdeckungsinformationen sowohl prozentual als auch numerisch aufgelistet. Enthält dieser Knoten eine nicht-triviale Bedingung, findet man hier auch die logische Struktur der Bedingung sowie alle Belegungen ihrer Atome tabellarisch aufgelistet. In der ersten Spalte befindet sich der Wahrheitsvektor für die gesamte

Bedingung, wie er aus der Testlogdatei eingelesen wurde. Die zweite Spalte enthält die Auswertung für die Gesamtbedingung und die darauffolgenden Spalten die Auswertung für alle atomaren Teilbedingungen. Für jede atomare Bedingung lässt sich das entsprechende MCDC-Paar, falls vorhanden, farblich hervorheben, indem man auf den entsprechenden Tabellenkopf der Bedingung klickt.

4.2.6 View *Coverage*

Die zentrale Auswertung der Überdeckung des Projektes erfolgt mittels der View *Coverage*. Hier werden analog zur View *Project* sämtliche Strukturen des Projektes hierarchisch in einem Baum aufgelistet, wobei die Darstellung der Baumstruktur analog zur View *Project* über das Viewmenü konfiguriert und somit auch zu einer flachen Darstellung gewechselt werden kann. Für jede einzelne Projektstruktur wird in den dazugehörigen Spalten die prozentuale Erfüllung der verschiedenen Überdeckungsmaße angegeben. Wenn die Überdeckungsanzeige
in der Toolbar aktiviert ist, werden die Maßzahlen zur besseren Übersichtlichkeit farbig hinterlegt, wobei die einzelnen Farben und Schranken in den Präferenzen definiert werden können.

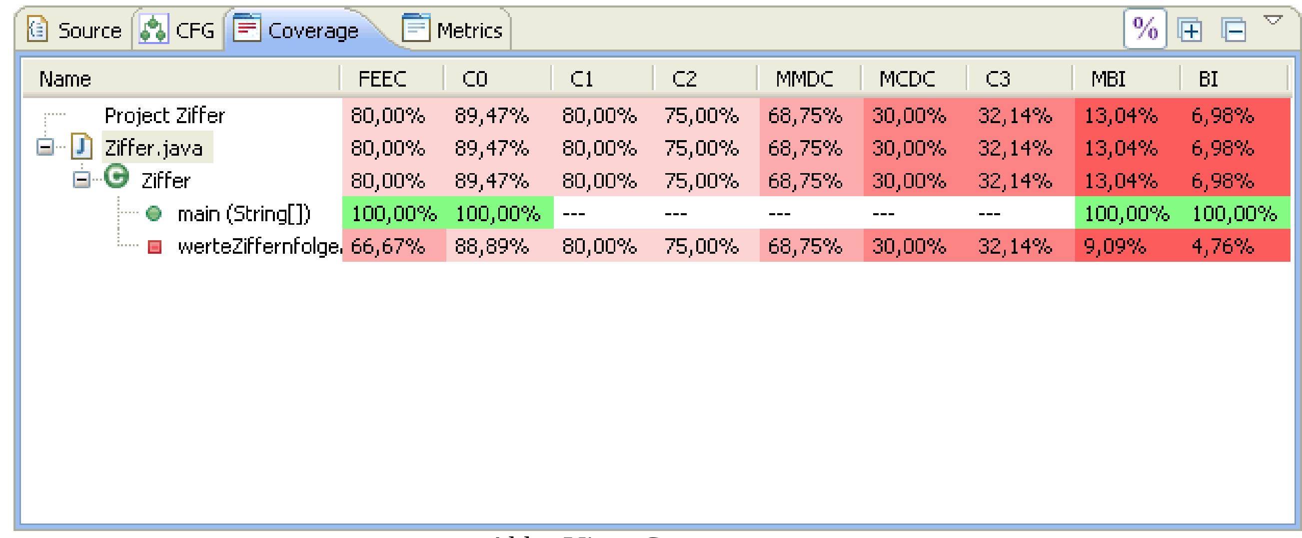
Die aufgeführten Überdeckungsmaße sind: Function Entry Exit Coverage (FEEC), Anweisungsüberdeckung (C0), Zweigüberdeckung (C1), einfache Bedingungsüberdeckung (C2), Minimale Mehrfach-Bedingungsüberdeckung (MMDC), Modifzierte Bedingungs-/Entscheidungsüberdeckung (MCDC), Mehrfach-Bedingungsüberdeckung (C3), Modifzierte Boundary-Interior-Pfadüberdeckung (MBI) sowie Boundary-Interior-Pfadüberdeckung(BI). Die Definition der einzelnen Überdeckungsmaße ist im Anhang aufgeführt.

Steht statt einer Wertangabe ein Strich in der View, so lässt sich das entsprechende Maß nicht auf diese Struktur anwenden, weil z.B. keine Bedingungen oder Anweisungen in dieser Klasse enthalten sind. Die Wertdarstellung selber lässt sich über das Ändern der Option
%
Change Info von der prozentualen in die Verhältnisdarstellung und wieder zurück ändern. Mit den beiden Buttons

Expand All und

Collapse All lässt sich in der hiearchischen Darstellung die Projektstruktur auf- bzw. zuklappen.

Die gesamte Tabelle lässt sich nach den einzelnen Spalten, d.h. nach Name und Überdeckungsgrad sortieren, indem man auf die entsprechenden Spaltenköpfe klickt. Die Sortierung erfolgt dabei ausschließlich auf der Basis der Wurzelelemente des Baumes, d.h. in der hierarchischen Darstellung auf den Wert der Dateien bzw. Klassen. Jedoch kann für eine Sortierung nach Funktionen über das Viewmenü in die flache Darstellung umgeschaltet werden.



Name	FEEC	C0	C1	C2	MMDC	MCDC	C3	MBI	BI
Project Ziffer	80,00%	89,47%	80,00%	75,00%	68,75%	30,00%	32,14%	13,04%	6,98%
Ziffer.java	80,00%	89,47%	80,00%	75,00%	68,75%	30,00%	32,14%	13,04%	6,98%
Ziffer	80,00%	89,47%	80,00%	75,00%	68,75%	30,00%	32,14%	13,04%	6,98%
main (String[])	100,00%	100,00%	---	---	---	---	---	100,00%	100,00%
werteZiffernfolge	66,67%	88,89%	80,00%	75,00%	68,75%	30,00%	32,14%	9,09%	4,76%

Abb.: View *Coverage*

4.2.7 View *Metrics*

Während der syntaktischen Analyse des Quellcodes beim Parsen werden eine Vielzahl statischer Metriken berechnet. In der View *Metrics* können diese direkt nach dem Einlesen des Projektes ausgewertet werden. Die View besteht analog zur View *Coverage* aus dem Projektbaum und einer Zuordnung folgender Maßzahlen zu den einzelnen Projekteinheiten: Zyklomatische Komplexität, Essentielle Komplexität, Lines of code, Anzahl der Anweisungen, Anzahl der Abzweigungen, Anzahl der Modifizierten Boundary-Interior-Pfadsegmente, Anzahl der Boundary-Interior-Pfade, Anzahl der Anweisungen mit Bedingungsauswertung, Anzahl der Atome in allen Bedingungen und Anzahl Bedingungen. Die Erläuterung und Definition der einzelnen Metriken ist im Anhang aufgeführt.

Auch hier lassen sich die Projektstrukturen wie in der View *Coverage* im Viewmenü in ihrer Darstellung konfigurieren und über die Buttons



Expand All und *Collapse All* auf- und zuklappen. Gleichermaßen funktioniert die Sortierung der Tabelle durch Anklicken der entsprechenden Spaltenköpfe.

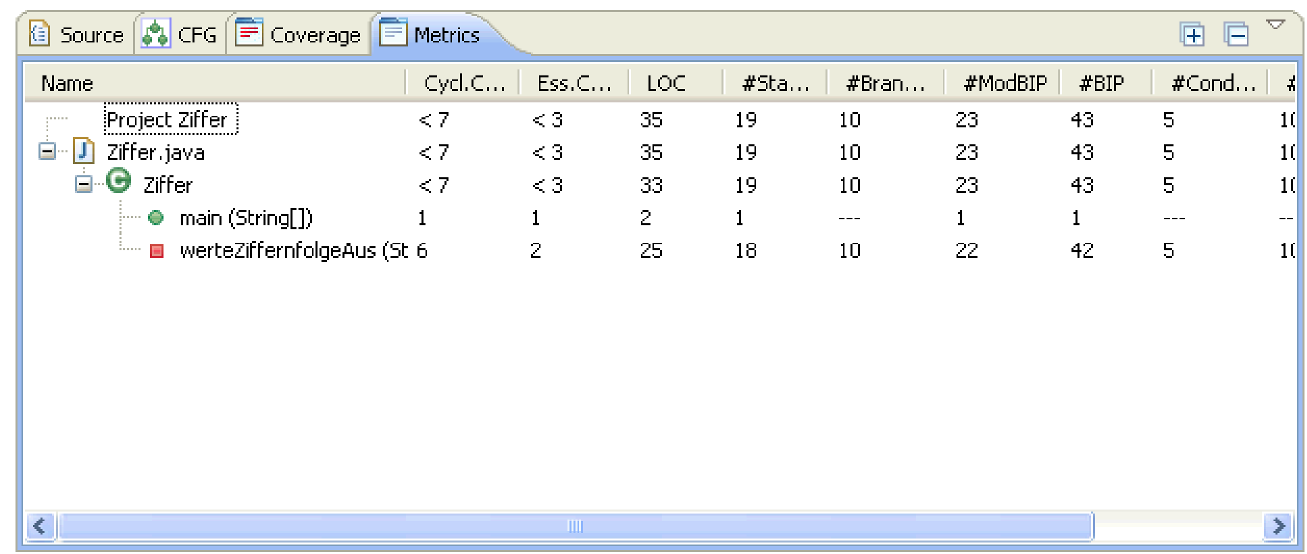


Abb.: View *Metrics*

4.3 Präferenzen

4.3.1 Präferenz View *CFG*

Auf der Präferenzseite zur View *CFG* kann man im ersten Optionsblock einstellen, welche

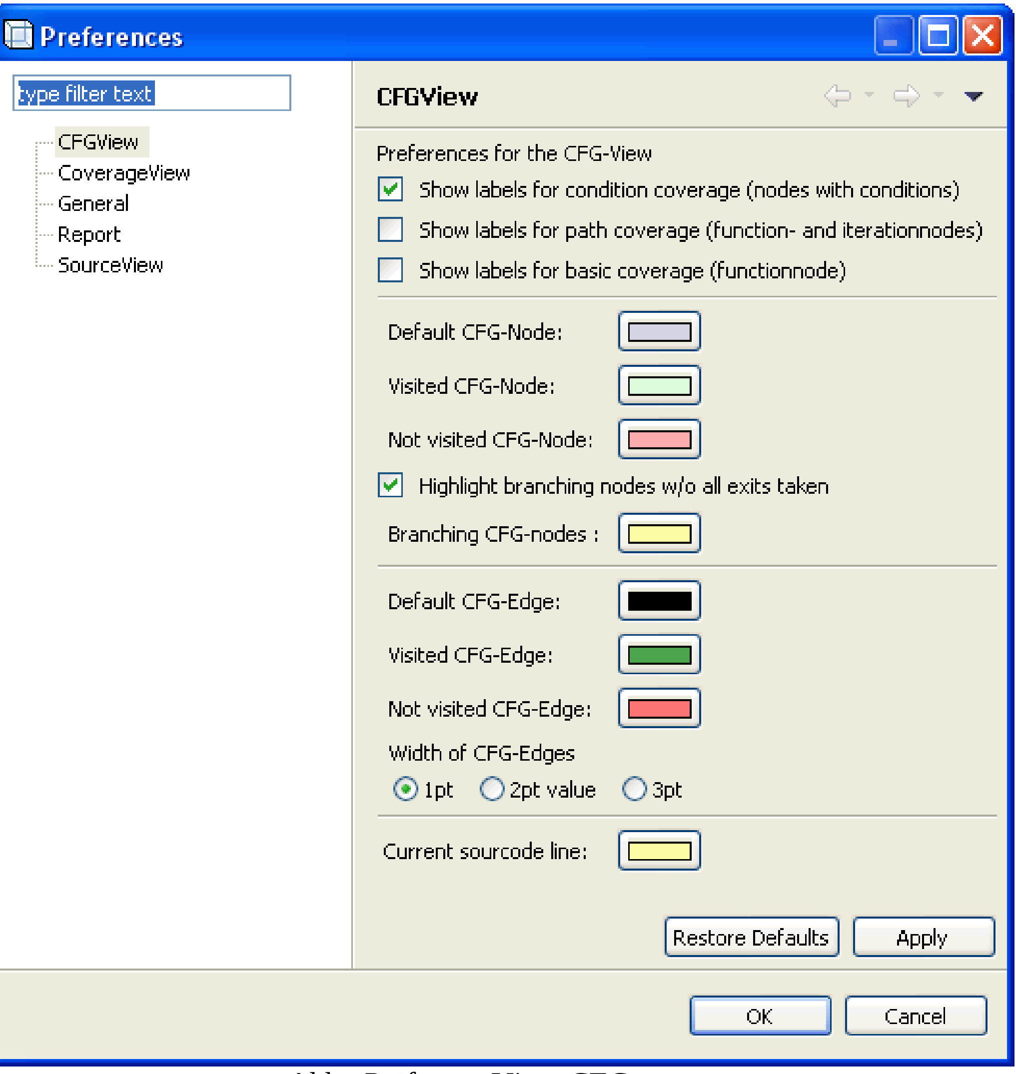


Abb.: Präferenz View *CFG*

Überdeckungsmaße im Kontrollflussgraphen durch kleine quadratische Label neben den Knoten angezeigt werden. Die Standardeinstellung zeigt nur die vier Label für die Bedingungsüberdeckungsmaße (C2, MMDC, MCDC, C3) an allen Knoten an, die nicht-triviale Bedingungen enthalten sowie am Funktionsknoten. Durch das Aktivieren der Anzeige von Pfadüberdeckungsmaßen (2. Option) erscheinen am Funktionsknoten drei weitere Label, welche die Modifizierte Boundary-Interior-Pfadüberdeckung insgesamt, die Überdeckung der Modifizierte Boundary-Interior-Pfade durch den Funktionsrumpf, sowie die Boundary-Interior-Pfadüberdeckung anzeigt. Analog zum zweiten Label wird auch an jedem Iterationsknoten ein Label für die Überdeckung der Modifizierte Boundary-Interior-Pfade für die entsprechende Iteration erstellt. Die letzte Option stellt die verbleibenden drei Überdeckungsmaße (FEEC, C0, C1) am Funktionsknoten dar.

Im zweiten Block lässt sich die Darstellung der Knoten des Kontrolflussgraphen konfigurieren. Es können Farben für die normalen Knoten ohne Überdeckungsanzeige, überdeckte und nichtüberdeckte Knoten ausgewählt werden. Für eine differenzierte Darstellung werden verzweigende Knoten, die zwar überdeckt, aber deren Ausgänge nicht vollständig überdeckt sind, farbig hervorgehoben. Deren Farbe lässt sich unter *Branching CFG-Nodes* einstellen. Durch das Entfernen des Häkchens an der Option darüber, lässt sich diese differenzierte Darstellung auch abstellen.

Dazu ergänzend lässt sich im dritten Block die Farbe der Kanten des Kontrollflussgraphen für die Darstellung ohne Überdeckung, sowie für überdeckte und nichtüberdeckte Kanten einstellen. Zusätzlich lässt sich die Linienstärke der Kanten auf einen Wert von eins bis drei einstellen.

>

Abschließen kann man noch die Farbe für die Hervorhebung derjenigen Zeile im Quelltext auswählen, in welcher sich der aktuelle ausgewählte Knoten im Kontrollflusgraphen befindet.

4.3.2 Präferenz View *Coverage*

Für die farbliche Hinterlegung der prozentualen

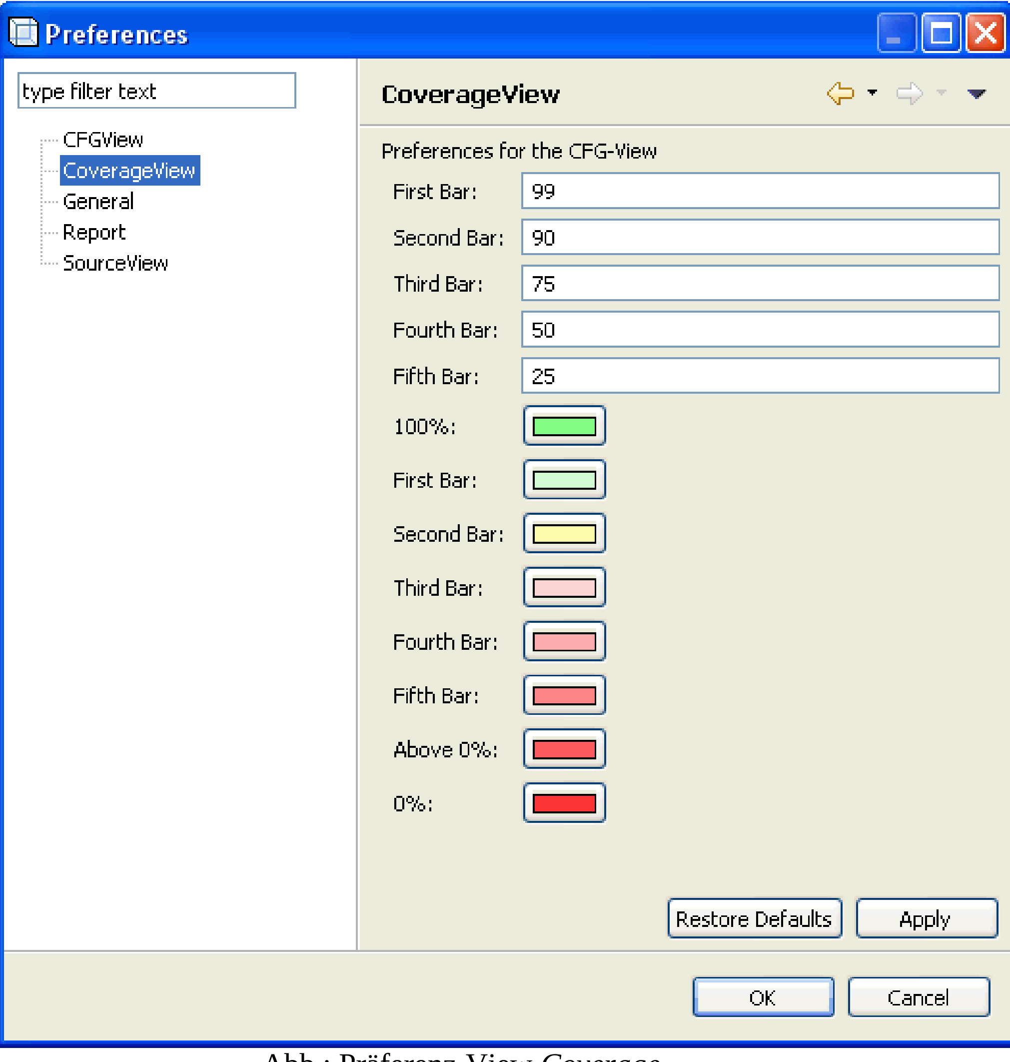


Abb.: Präferenz View *Coverage*

Überdeckungsanzeige in der View *Coverage* lassen sich hier Schranken und Farben definieren.

Zwei natürliche Schranken sind dabei eine vollständige (100%) und überhaupt keine (0%) Überdeckung. Für fünf weitere Schranken lassen sich prozentuale Werte eingeben, so dass beim Erreichen des Wertes die entsprechende Zelle mit der dieser Schranke zugewiesenen Farbe hinterlegt wird. Die default-Schranken sind hierfür: 25%, 50%, 75%, 90% und 99%. Falls für eine Schranke ein Wert angegeben wird, der größer ist als einer der über ihr liegenden Schranken, so wird er bei der Farbbestimmung ignoriert.

Anschließend lässt sich jeder Schranke ein Farbe zuordnen, mit welcher in der Tabelle der View *Coverage* alle Zellen hinterlegt werden, in welchen die prozentuale Überdeckung diese Schranke erreicht.

4.3.3 Präferenz *General*

Allgemeine Einstellungen zu SOTA lassen sich im

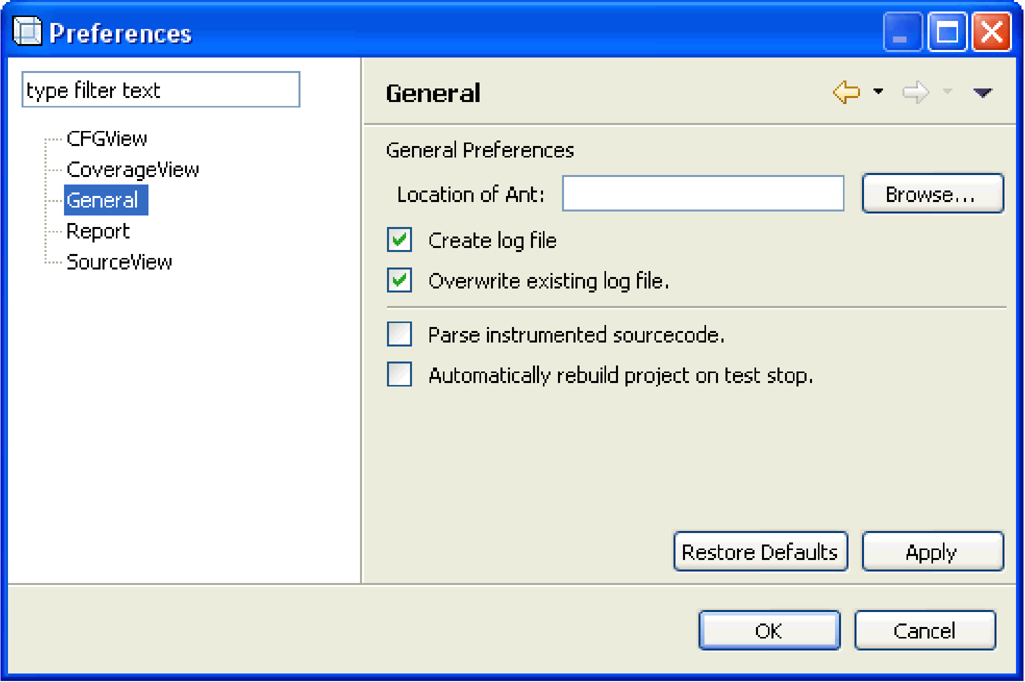


Abb.: Präferenz *General*

Präferenzenpunkt *General* vornehmen. Falls man für ein Projekt ein Ant-Buildfile zur automatischen Kompilation nutzen möchte, so muss man hier die entsprechende Ant-Datei (\bin\ant.bat) einer installierte Ant-Version einbinden. Daraufhin lässt sich im Dialog *Start Test*

die Option *Build Project* auswählen und auch der Menüeintrag *Build Project*

bzw. sein Pendant in der Toolbar ist aktiviert. Wurde das Testprogramm mit Eclipse erstellt, so kann ein Ant-Buildfile für die Kompilation des Programmes dort über *File -> Export -> Ant Buildfile* exportiert werden.

Wird *Create log file* aktiviert, so werden für jeden Programmstart Systemmeldungen aus SOTA in Logdateien im Format *sota_<YY-MM-DD>_<index>.log* gesichert. Ist zusätzlich die Option *Overwrite existing log file* aktiviert, so wird immer nur eine Logdatei namens *sota.log* erstellt und bei jedem neuen Programmstart überschrieben.

Die beiden letzten Optionen bestimmmen noch allgemeine Teilaspekte des Verhaltens von SOTA. Ist *Parse instrumented sourcecode* aktiviert, so wird das Projek nach dem Teststart neu geparst, die Ansicht von Überdeckungsinformationen wird deaktiviert und der Darstellung des Projektes in allen Views liegen die nun instrumentierten Quellen zugrunde. Der dargestellte Quellcode in der View *Source* ist in diesem Fall immer identisch mit den aktuellen Quellen, d.h. die beiden Statusanzeigen in der Statuszeile weisen immer den gleichen Wert auf. Ist die Option deaktiviert, so wird statt der instrumentierten Datei das Backup geparst und angezeigt. Somit können unabhängig vom Status der Quelldateien auch immer Testlogs ausgewertet werden. Die letzte Option bestimmt, ob beim Teststopp zusätzlich zur Wiederherstellung des originalen Quellcodes das Testprojekt auch automatisch neu kompiliert werden soll, falls ein entsprechendes Ant-Skript eingebunden wurde. Andernfalls würden die Binaries des Testprogramm weiterhin instrumentiert bleiben und beständig Logausschriften erzeugen.

4.3.4 Präferenz *Report*

Im Präferenzpunkt *Report* lässt sich die Ausgabe des Reports in die html-Datei konfigurieren.

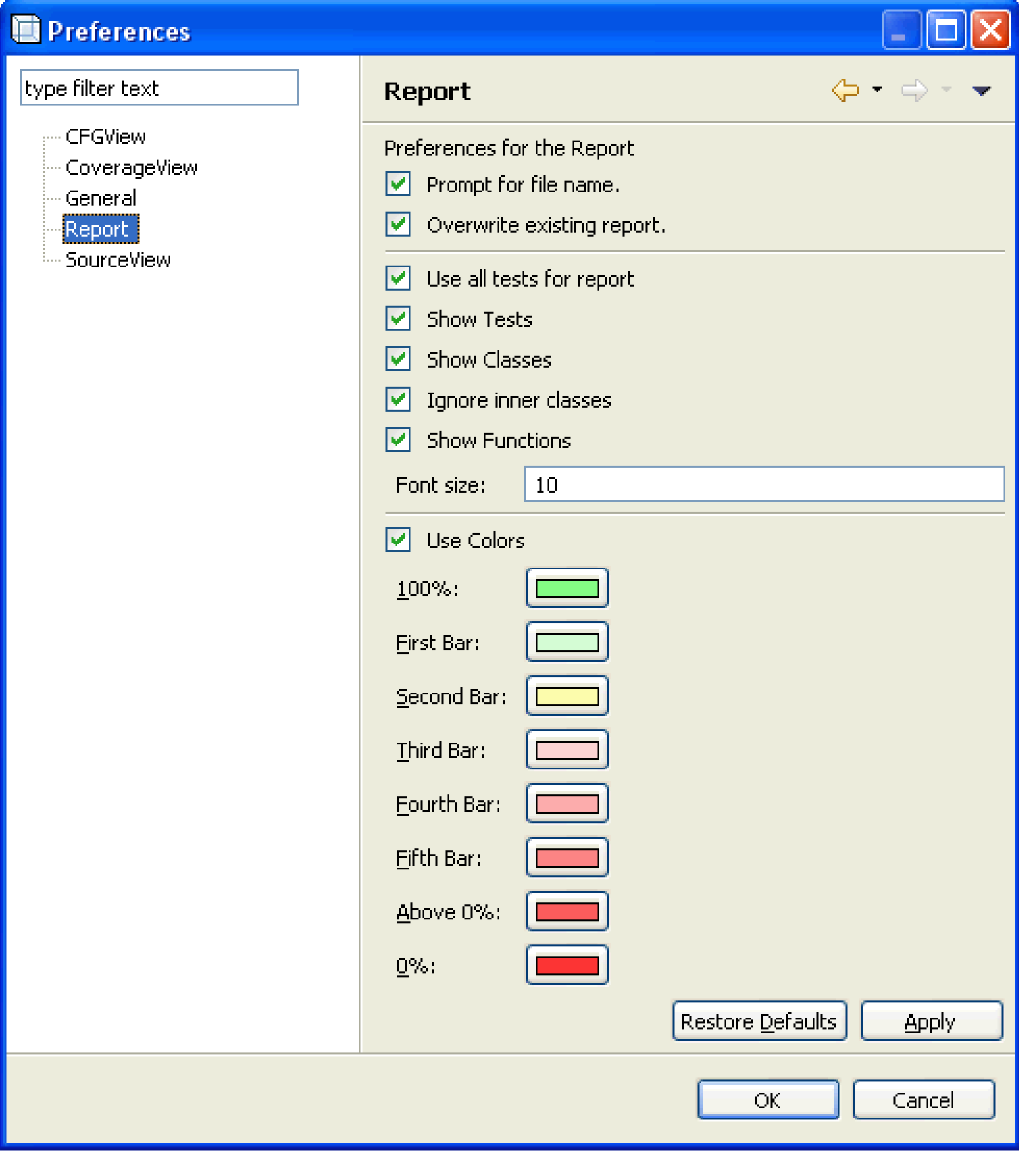


Abb.: Präferenz *Report*

Ist im ersten Block die Option *Prompt for file name* aktiviert, öffnet sich bei Wahl des Menüpunktes *Create Report* ein Dateiauswahldialog, der nach dem Namen der zu erstellenden Reportdatei fragt. Andernfalls wird der Report unter *report.html* im Projektverzeichnis des Testprogrammes erstellt und bei jeder neuen Reporterstellung überschrieben, oder aber jeweils ein neuer Name für den Report nach dem Schema *report_<date>_<index>.html* erstellt. Dieses Verhalten wird über den Präferenzpunkt *Overwrite existing report* bestimmt.

Der zweite Block bestimmt den Inhalt und die Darstellung des Reports. So kann eingestellt werden, ob alle importierten Testlogs für den Report genutzt werden sollen, oder nur die aktuell ausgewählten, und welche der folgenden Elemente in der Report-Datei aufgeführt werden sollen: die verwendeten Tests mit IScheme und Beschreibung, eine Übersicht über die Überdeckung aller Klassen (inklusive oder exklusive innerer Klassen) und/oder eine Übersicht über alle Funktionen, geordnet nach ihren Klassen. Schließlich kann noch die Font-Größe für das Reportfile vorgegeben werden.

Ist *Use Colors* ausgewählt, werden die Überdeckungsmaßzahlen in der Reportdatei analog zur Darstellung in der View *Coverage* entsprechend dem Grad der Überdeckung farblich hinterlegt. Die Werte für die Schranken werden aus dem Präferenzpunkt View *Coverage* übernommen, die Farben jeder einzelnen Schranke kann separat für die Reportdatei an dieser Stelle definiert werden.

4.3.5 Präferenz View *Source*

Das

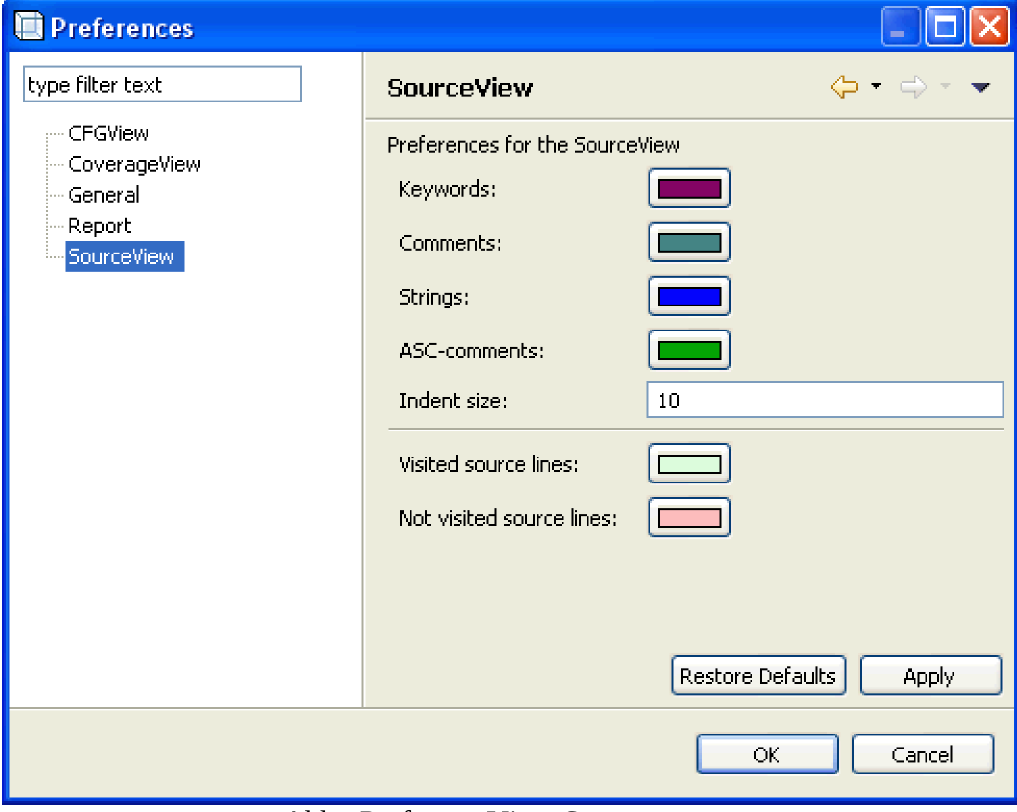


Abb.: Präferenz View *Source*

Syntaxhighlighting in der View *Source* lässt sich im gleichnamigen Präferenzpunkt einstellen. Die Farben für Keywords der Sprache, Kommentare, Strings sowie jene Kommentare, die durch das Instrumentieren durch SOTA hinzugefügt wurden, kann man hier frei wählen und schließlich auch die Schriftgröße für die Darstellung des Quelltextes einstellen

Für die Darstellung der Überdeckung des Quellcodes in der View *Source* kann man hier die Hintergrundfarbe für die überdeckten und nicht-überdeckten Quellcodezeilen wählen. Diese farbige Hinterlegung erfolgt, sobald Testslogs eingelesen wurden und die Option *Show Coverage* aktiviert ist.

4.4 Projekt löschen

Eine Funktion "Delete Project" ist nicht implementiert. D.h., dass alle zu einem Projekt gehörenden SOTA-Dateien mit Ausnahme der Logdateien (siehe [View *Testlogs*]) von Hand gelöscht werden müssen.

Alle SOTA-Dateien sind in 5.1beschrieben. Die von SOTA angelegten und zur Bereinigung des Systems zu löschenden Dateien sind im einzelnen:

- SOTA Basisverzeichnis:
  - <projectname>.project
- Basisverzeichnis des Testprogrammes (*Project directory*)
  - <xyz>.java.backup
  - report.html
  - report_<date>_<index>.html
  - \lib\ASCLogger.jar
- Ausführungsverzeichnis des Testprogrammes (*Execution directory*)
  - <testname>.log
  - ASCLogger.ini

Vor dem Löschen der Backup-Dateien ist es unbedingt nötig, etwaige instrumentierte Quelldateien wieder in ihren Originalzustand zu überführen (über den Menüpunkt *Restore Sources*), da dies ohne Backup-Datei kaum mehr möglich ist!



SOTA Benutzer-Dokumentation - Version 1.0

5 Dateien

5.1 Übersicht

SOTA-Verzeichnis

Das SOTA-Verzeichnis beinhaltet das SOTA-System, wie z.B. die Startdatei SOTA.exe und die mit SOTA installierte Eclipse-Rich-Client-Platform, sowie weitere spezielle SOTA-Dateien und die Projektdateien.

sota.log, sota_<datum>_<index>.log	Hier protokolliert SOTA alle seine Aktivitäten.
language.spec	Die Spezifikationsdatei für die verschiedenen unterstützten Sprachen.
ASCLogger.jar	Die Logging-Komponente für Java-Testprogramme muss in das Testprogramm eingebunden werden und verlangt beim Test nach einer Initialisierungsdatei namens *ASCLogger.ini* (siehe unten).
<projectname>.project	Für jedes Projekt werden hier allgemeine Projektdaten vermerkt.
SOTA-ATM.jar	Das Automatische Test-Modul von SOTA, womit dessen Testfunktionalität ohne GUI einsetzbar ist. *SOTA-ATM.jar* ist eine ausführbare Jar-Datei, die aber auch als Programmbibliothek importiert werden kann. Der Zugriff ist damit sowohl über die Kommandozeile (Skripte) als auch softwaretechnisch möglich. Siehe dazu das Tutorium.

Basisverzeichnis des Testprogramms (*Project directory*)

Das Basisverzeichnis des Projektes beinhaltet (eventuell in einem Unterverzeichnis) die Quellen des Testprogrammes und an gleicher Stelle auch deren Backups, die von SOTA beim Erzeugen des Projektes erstellt werden. An diese Stelle werden auch die von SOTA automatisch generierten Reportdateien geschrieben, wenn nicht in den Präferenzen die Nachfrage über einen Dateiauswahldialog aktiviert wurde.

(\src\) <xyz>.java	Die Quelldateien des Testprojektes, sind nach dem Teststart (teilweise) instrumentiert.

(\src\) <xyz>.java.backup	Das Backup der originalen Quelldateien wird vor dem Teststart für alle nicht-instrumentierten Quelldateien erstellt.
(\bin\)<xyz>.class	Die kompilierten Class-Dateien, die je nach Zustand der Quelldateien instrumentiert sein können.
report.html, report_<datum>_<index>.html	Die von SOTA generierten Reportdateien.
<antbuildfilename>.xml	Eine eventuell vorhandenes Ant-Buildfile, das die Kompilation des Testprogrammes aus SOTA heraus ermöglicht. Kann bei Eclipse-Projekten über *File -> Export -> Ant Buildfile* erzeugt werden.
<runscriptname>.bat	Eine Batchdatei zum Starten des Testprogrammes, die, wenn in SOTA eingebunden, in Zusammenspiel mit dem Andbuildscript den manuellen Programmtest aus SOTA heraus ermöglicht.

Ausführungsverzeichnis des Testprogramms (*Execution directory*)

Das Ausführungsverzeichnis ist das Verzeichnis, aus welchem das Testprogramm gestartet wird. In den meisten Fällen entspricht diese Verzeichnis dem Basisverzeichnis des Testprogrammes. Eine Ausnahme hierzu bildet z.B. das Testen einer Eclipse-RCP-Application durch Eclipse, da hier das Ausführungsverzeichnis dem Basisverzeichnis der genutzten Rich-Client-Platform entspricht, d.h. im Allgemeinen: ..\eclipse\ .

ASCLogger.ini	Diese Initialisierungsdatei für die Loggingkomponente wird beim Teststart in dieses Verzeichnis kopiert, von wo es durch die Klasse ASCLogger.jar für die Erstellung der Logdatei benötigt wird.
<testname>.log	Die durch den ASCLogger erstellten Logdateien.

5.2 Projektdatei

Die Nutzung von SOTA erfordert es nicht, die Projektdateien zu verändern. Möchte man aber SOTA-ATM als automatisches Testmodul einsetzen, kann es von Vorteil sein, Projektdateien manuell oder per Skript zu erstellen bzw. zu verändern, um eine umfangreiche Kontrolle über den Test zu haben.

Die von SOTA verwendeten Projektdateien sind einfache XML-Dateien, welche die projektspezifischen Informationen als Werte der einzelnen Entitäten speichern. Ihre Form wird durch die Schema-Definition project.dtd definiert ist

Die Projektdatei definiert ein Project, das mindestens durch folgende Werte charakterisiert wird:

Name	der Name des Projektes, muss identisch mit dem Dateinamen ohne Endung sein.
Language	die Sprache des Projektes, muss in der Sprachspezifikation aufgeführt sein.
Prefix	der Präfix, der beim Instrumentieren zur Kennzeichnung von SOTA eingeführter Variablen dient, um Namenskollisionen zu vermeiden.
BackupExtension	die Endung, mit welcher die Backup-Dateien gespeichert werden sollen.
ProjectDir	das Basisverzeichnis des Projektes.
ExecDir	das Ausführungsverzeichnis des Projektes.
SourceFiles	eine Auflistung der Quelldateien (als SourceFile), die zum Projekt gehören.

Optional ist die Verwendung folgender Werte, welche spezielle Features des Programmes ermöglichen:

AntLocation	der Pfad zur Apache-Ant-Installation, notwendig für das automatische Kompilieren des Projektes.
AntBuildFile	die Ant-Builddatei, über welche das Projekt kompiliert werden kann.
RunScript	das Skript, welches zum Start des Projektes genutzt werden soll.
ISchemes	eine Auflistung an InstrumentationsSchemata (als IScheme), die die variable Instrumentierung des Projektes ermöglichen. Ein IScheme besteht dabei aus dem Namen und einer Zuordnung der einzelnen Strukturen des Projektes (Datei, Klasse, Funktion) zu einem Instrumentierungslevel, sowie einer etwaigen Beschreibung.

Es folgt eine exemplarische Projektdatei für das Projekt Ziffer, welche auch ein IScheme definiert, dass die einzige Quellcodedatei nach Level 1 und ihre Methode "werteZiffernfolgeAus" nach Level 2 instrumentier.

```
<Project>
   <Name>Ziffer</Name>
   <Language>Java</Language>
   <Prefix>asc</Prefix>
   <BackupExtension>backup</BackupExtension>
   <ProjectDir>D:\Development\workspace\Ziffer</ProjectDir>
   <ExecDir>D:\Development\workspace\Ziffer</ExecDir>
   <SourceFiles>
      <File>D:\Development\workspace\Ziffer\src\Ziffer.java</File>
   </SourceFiles>
   <ISchemes>
      <IScheme>
         <Name>Schema F</Name>
         <Description>Ziffer.java Lvl1, werteZiffernfolgeAus Lvl2</Description>
         <Level1>
            <Item>Ziffer.java</Item>
         </Level1>
         <Level2>
            <Item>Ziffer.java:Ziffer::werteZiffernfolgeAus(String)</Item>
         </Level2>
      </IScheme>
   </ISchemes>
</Project>
```

5.3 Reportdatei

Es folgt ein Beispielreport für das Projekt Ziffer bei den Eingabewerten "..", ".2", "1", "1.1" und ohne Eingabewert. (Zum Programm siehe 6.1.1.)

Jede Reportdatei führt zu Beginn den Names des Projektes und den Zeitpunkt der Erstellung auf. Es folgt eine Auflistung aller Überdeckungsmaße samt den im Test erreichten Werten für dieses Projekt, sowie einzelne statische Maße. Die weiteren Tabellen der Reportdatei werden nur erstellt, wenn diese in den Präferenzen ausgewählt wurden. Die Standardeinstellung gibt alle Tabellen aus.

Wurde *Show Testlogs* ausgewählt, folgt eine Auflistung aller Testdateien mit dem entsprechenden IScheme, sowie ihrer Beschreibung, die für diesen Report genutzt wurden. Wurde in den Präferenzen *Use all tests for report* aktiviert, werden alle importierten Testlogs für den Report genutzt und hier aufgeführt.

Bei aktivierter *Show Classes* Option folgt eine Tabelle, die zusätzlich zum Gesamtprojekt alle Klassen des Projektes auflistet und ihnen in einer Tabelle ihre Überdecksmaße gegenüberstellt, welche je nach Wert farblich hinterlegt sein können (vgl. Präferenzen).

Ist *Show Functions* aktiviert, wird nach dem Punkt *Detailed Coverage* für jede Klasse nun eine Tabelle mit den Überdeckungsmaßen der Klasse selbst und nachfolgend aller ihrer Funktionen in die Reportdatei geschrieben. Es führt ein Link von jeder Klasse in der Tabelle *Coverage of Classes* zu der entsprechenden Auflistung ihrer Funktionen in dem Abschnitt *Detailed Coverage*.

Ein umfangreicherer Beispielreport für das Projekt HUSemOrg lieg der Benutzerdokumenation bei.

SOTA Coverage Report

Project: Ziffer

created: 2009-03-23 13:54:35

Function Entry-Exit Coverage (FEEC)	100,00%
Statement Coverage (C0)	100,00%
Decision Coverage (C1)	100,00%
Condition Coverage (C2)	95,00%
Minimal Multiple Decision Coverage (MMDC)	93,75%
Modified Condition Decision Coverage (MCDC)	50,00%
Multiple Condition Coverage (C3)	46,43%
Modified Boundary-Interior Path Coverage (ModBI)	26,09%
Boundary-Interior Path Coverage (BI)	26,09%

# Files	1
# Classes (TopLevel- + inner Classes)	1 (1 + 0)
# Functions	2
# Lines	35
# Statements	19
# Conditions	16

Tests

test ..	
Level 2 instrumentation	
test .2	
Level 2 instrumentation	
test 1	
Level 2 instrumentation	
test 1.1	
Level 2 instrumentation	
test empty	
Level 2 instrumentation	

Coverage of Classes

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Project Ziffer	100,00	100,00	100,00	95,00	93,75	50,00	46,43	26,09	13,95
Class Ziffer	100,00	100,00	100,00	95,00	93,75	50,00	46,43	26,09	13,95

Detailed Coverage

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class Ziffer	100,00	100,00	100,00	95,00	93,75	50,00	46,43	26,09	13,95
- main (String[])	100,00	100,00	---	---	---	---	---	100,00	100,00
- werteZiffernfolgeAus (String)	100,00	100,00	100,00	95,00	93,75	50,00	46,43	22,73	11,90

SOTA Benutzer-Dokumentation - Version 1.0

6 Tutorien

Zur Erläuterung der Arbeitsweise mit SOTA behandeln die folgenden drei Tutorien die drei Einsatzmöglichkeiten von SOTA im Manuellen Programmtes, dem Programmtest mit einem externen Testprogramm und dem Test als Bibliothek in einem automatischen Testsystem.

6.1 Manueller Programmtest - Ziffernprogramm

6.1.1 Ziffernprogramm

Basis für die Tutorien zum Manuellen Programmtes ist ein einfaches Java-Programm, welches versucht, aus einem String eine positivrationale Zahl zu lesen. Das Testprogramm besteht aus einer Klasse *Ziffer* mit einer *main*-Funktion und der Funktion *werteZiffernfolgeAus*, welche die Auswertung des Strings übernimmt.

Der String kann entweder als Parameter des Programmes übergeben oder im Quellcode vorgegeben werden. Der im Quellcode "hartverdrahtete" String wird ausgewertet, wenn das Programm parameterlos aufgerufen wird. Das Programm gibt die Ziffer aus, wenn das Auslesen erfolgreich war, oder "-1", falls ein Fehler aufgetreten ist, d.h. der String keine solche Zahl enthielt.

```
public class Ziffer {

   public static void main(String[] args) {

      if(args.length==0)
         System.out.println(werteZiffernfolgeAus("."));
      else
         System.out.println(werteZiffernfolgeAus(args[0]));
```

```
            }
            private static double werteZiffernfolgeAus(String inZiffernString) {

               double wert = 0.0;
               double genauigkeit = 1.0;
               String woBinIch = "VorKomma";
               boolean fehlerfrei = true;
               int position = 1;

               while(position <= inZiffernString.length() &fehlerfrei) {

                  String zchn = inZiffernString.substring(position-1, position);

                  if(zchn.matches("[0-9]")) {

                     if(woBinIch.equals("NachKomma"))
                        genauigkeit = genauigkeit / 10.0;
                        wert = 10.0*wert + Double.parseDouble(zchn);
                  }
                  else if(zchn.equals(".")  & woBinIch.equals("VorKomma"))
                     woBinIch  = "NachKomma";
                  else
                     fehlerfrei = false;

                  position ++;

               }

               if(!fehlerfrei | inZiffernString.length()==0 |
 ((woBinIch.equals("NachKomma")&inZiffernString.length()==1)))
                  return -1.0;
               else
                  return wert*genauigkeit;
             }
         }
```

6.1.2 SOTA und Eclipse

Allgemeiner Ablauf

Das Ziffernprogramm soll in Eclipse erstellt und dann einem strukturorientierten Programmtest unterworfen werden. Dies wird durch eine Folge von vier Phasen realisiert:

1. Eclipse: Programmeingabe
2. SOTA: Vorbereitungsphase
3. Eclipse: Testphase
4. SOTA: Auswertungsphase.

Das nachfolgende Datenflussdiagramm gibt alle wichtigen Aktionen sowie Ein-/Ausgaben wieder. Die vier Phasen ergeben sich durch den Wechsel zwischen Eclipse und SOTA und sind farblich differenziert dargestellt. Die Schnittstelle zwischen Eclipse und SOTA wird ausschließlich über die angegebenen Dateien realisiert.

Abb.: DFD-Diagramm *Manueller Test mit Eclipse*

Detailablauf

1. Eclipse: Programmeingabe

Das Testprogramm soll mit Eclipse erstellt werden. Falls Eclipse noch nicht installiert ist, kann dies anhand folgender Anleitung vollzogen werden: http://wiki.eclipse.org/Eclipse/Installation.

In Eclipse legt man zuallererst ein neues Projekt für das Testprogramm an. Dies lässt sich im Menü über *File* -> *New* ->

Java Project erstellen. In dem sich öffnenden Dialog gibt man als Namen für das Projekt "Ziffer" ein, alle anderen Optionen können so verbleiben und der Wizard kann schon auf der ersten Seite durch *Finish* abgeschlossen werden.

Für das angelegte Projekt erstellt man nun eine Klasse *Ziffer*, indem man über *File* -> *New* ->

Class den entsprechendenn Wizard aufruft. Hier trägt man den Namen der Klasse - *Ziffer* - ein und beendet den Dialog. In die erstelle Java-Datei, die sich im Hauptfenster von Eclipse öffnet, kopiert man abschließend den oben aufgelisteten Quellcode.

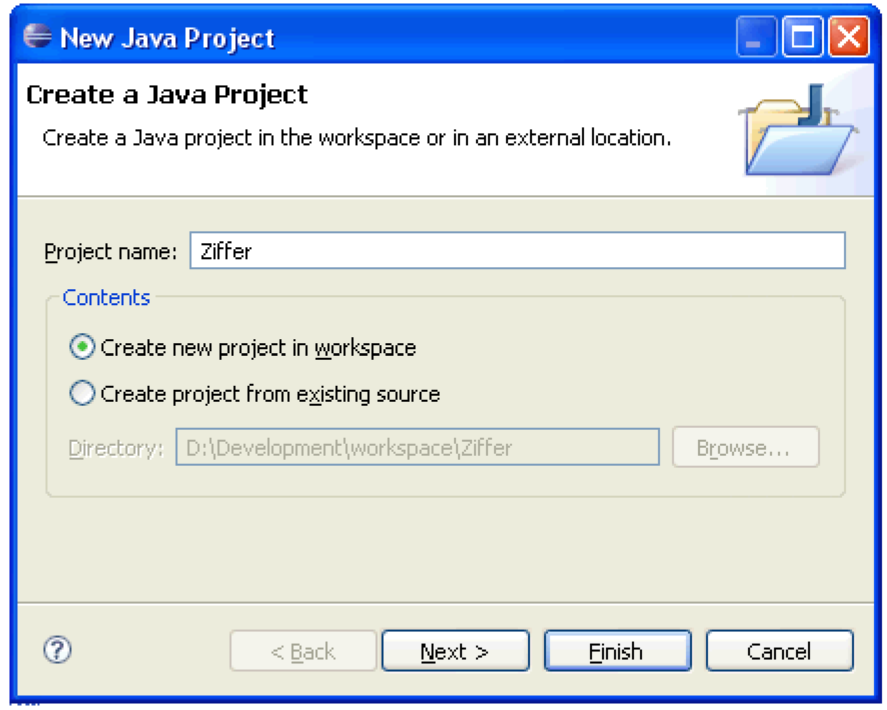


Abb.: Eclipse - *New Java Project*

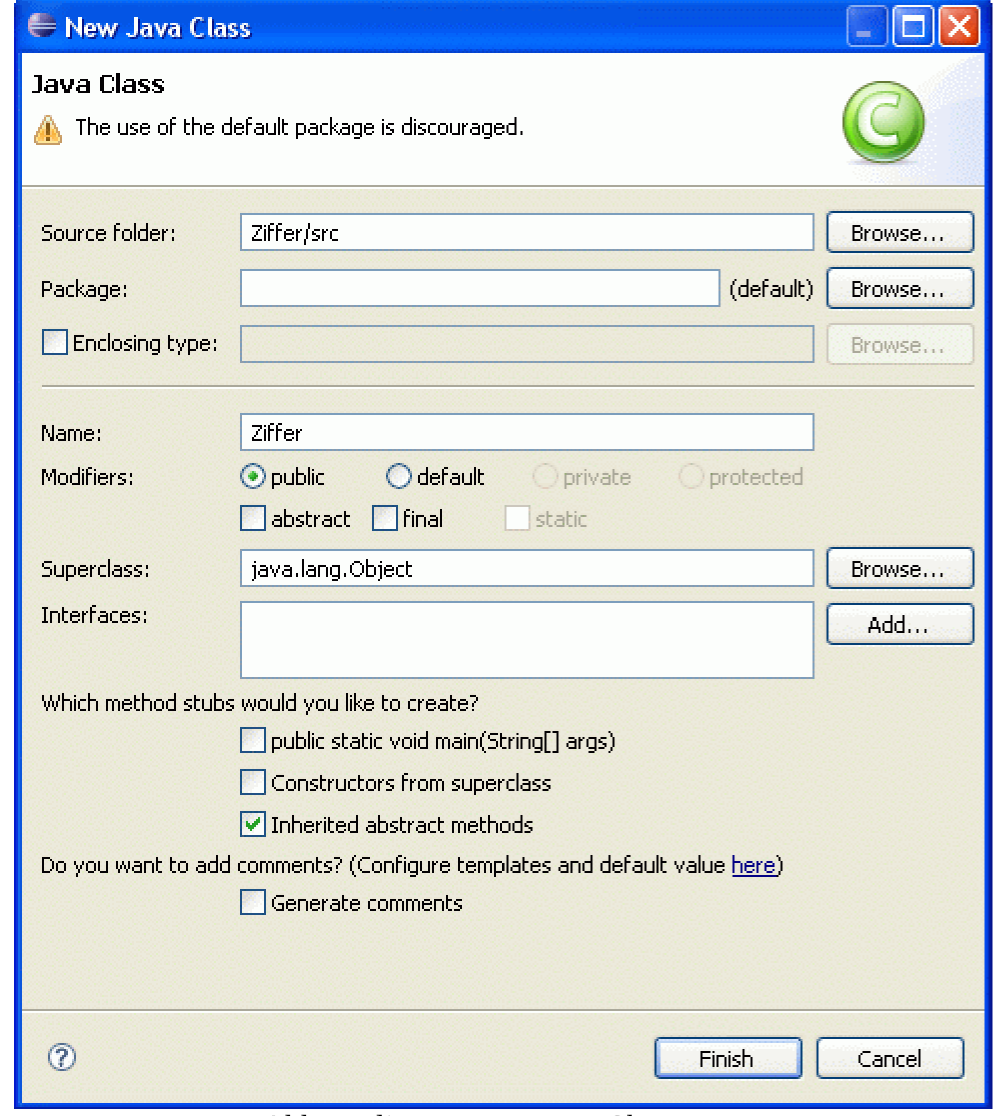


Abb.: Eclipse - *New Java Class*

Als nächstes ist es noch notwendig, dem Projekt die Loggingkomponente als Bibliothek

hinzuzufügen. Dazu erstellt man im Basisverzeichnis des Projektes (..\workspace\Ziffer\) einen Ordner *lib* und kopiert die Datei *ASCLogger.jar* aus dem SOTA-Verzeichnis dorthin. Wenn man dann die Eclipse-Projektübersicht aktualisiert (F5), so taucht diese Datei samt Verzeichnis dort auf (siehe Abbildung). Jetzt ist sie nur noch in den Buildpath des Projektes einzutragen, wozu man einfach per rechter Maustaste auf der ASCLogger.jar im Kontextmenü *Build Path* ->
Add to Build Path auswählt. Daraufhin wird die Bibliothek bei den *Referenced Libraries* aufgenommen. Damit ist die Programmerstellung und die Vorbereitung für die Instrumentierung des Programmes durch SOTA abgeschossen.

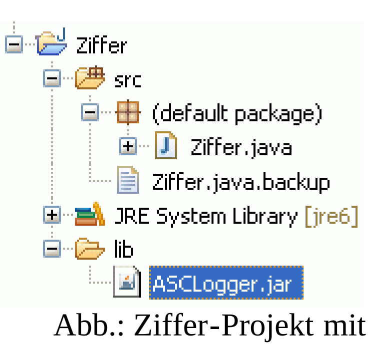


Abb.: Ziffer-Projekt mit ASCLogger.jar ...

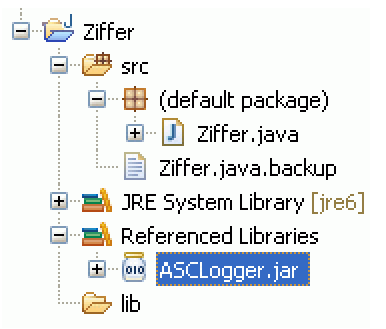


Abb.: ... und mit ASCLogger.jar im Buildpath

2. SOTA: Vorbereitungsphase - Projekterstellung

Nach dem Starten von SOTA über die SOTA.exe ist für das Testprogramm ein Projekt in SOTA anzulegen. Über den Menüpunkt

New Project öffnet sich ein zweiseitiger Wizard, der durch die Projekterstellung führt.

Auf der ersten Seite ist der Name des Projektes - Ziffer - sowie das Basisverzeichnis (*Project directory*) des in Punkt 1 erstellten Projektes einzugegeben. Das Ausführungsverzeichns (*Execution directory*) des Projektes wird von SOTA automatisch auf das gleiche Verzeichnis gesetzt und muss nicht mehr verändert werden, da bei diesem Projekt beide Verzeichnisse identisch sind. Damit ist die erste Seite fertig ausgefüllt und über den *Next*-Button kann zur zweiten Seite des Wizards navigiert werden.

Auf der zweiten Seite sind die Quellen des Projektes zu importieren. Beim Ziffernprojekt existiert nur eine einzige Datei, das Markieren des Basisverzeichnisses genügt. Mit dem *Finish*-Button kann der Wizard beendet werden, woraufhin die Quelldatei eingelesen und geparst wird.

Mit dem Speichern des Projektes ist die Projekterstellung vollzogen. Es kann nun beim nächsten Mal über den Menüpunkt

Open Project geladen werden. Sofort nach der Projekerstellung bzw. dem Laden eines Projektes kann der Quellcode in der View *Source* und der Kontrollflussgraph einer jeden Funktion (sobald sie in der View *Project* ausgewählt wurde) in der View *CFG* eingesehen werden. Die beim Parsen der Quellen berechneten statischen Maße sind in der View *Metrics* aufgelistet.

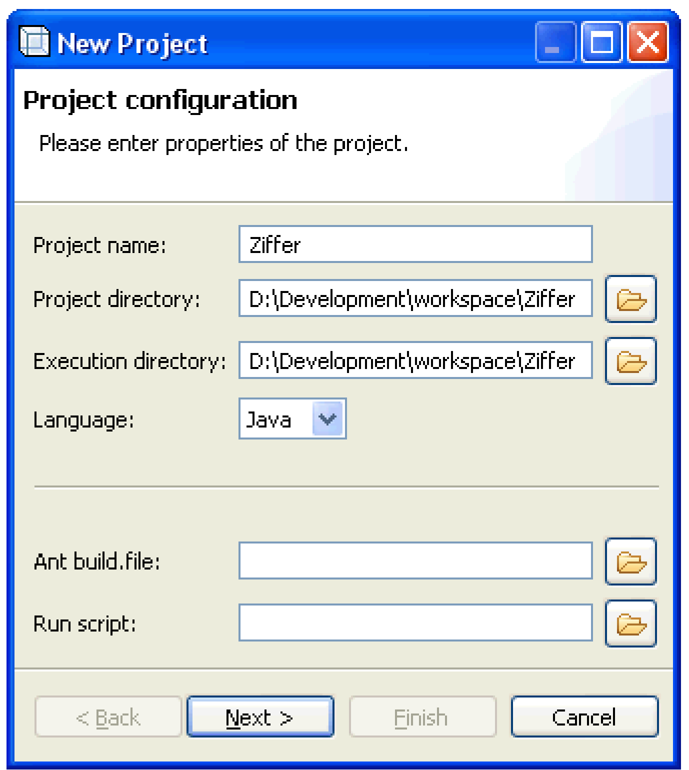


Abb.: erste Wizardseite

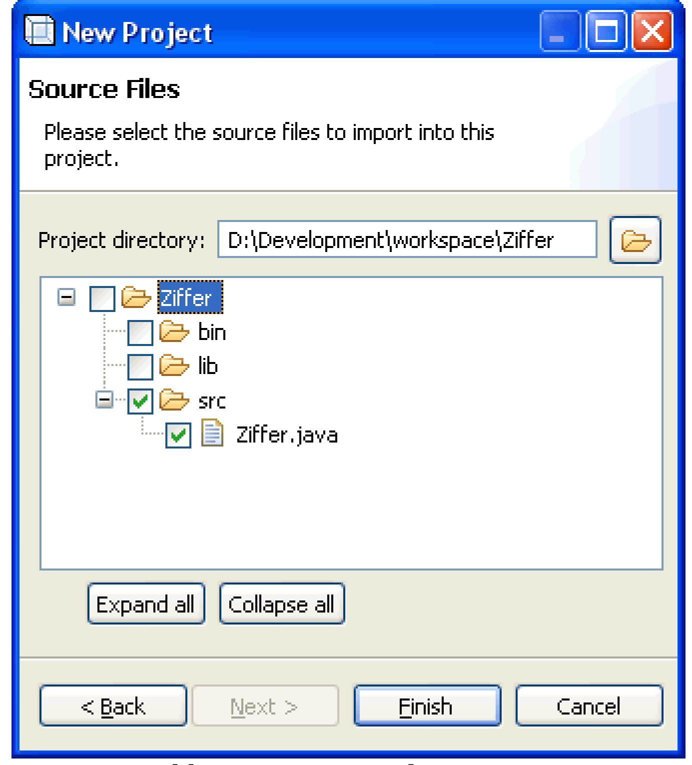


Abb.: zweite Wizardseite

3. SOTA: Vorbereitungsphase - Instrumentierung

Als nächster Schritt ist das Projekt für den nächsten Testlauf zu instrumentieren, d.h. mit Anweisungen anzureichern, die während

des Programmablaufes Daten in eine Logdatei schreiben, die seine vollständige Rekonstruktion ermöglichen. Dazu wählt man im Menü *Start Test*. Im sich öffnenden Dialog ist eine Name für den Test einzugeben und ein Instrumentationsschema auszuwählen. Der Testname bestimmt auch den Namen der Logdatei, unter welcher die Logdaten gespeichert werden. Das Instrumentationsschema gibt die Art der Instrumentierung für alle Strukturen des Projektes vor. Hier ist im Normalfall das IScheme *Level 2 instrumentation* zu wählen, welches eine minimale Instrumentierung zur Berechnung aller Überdeckungsmaße für alle Dateien vollzieht.

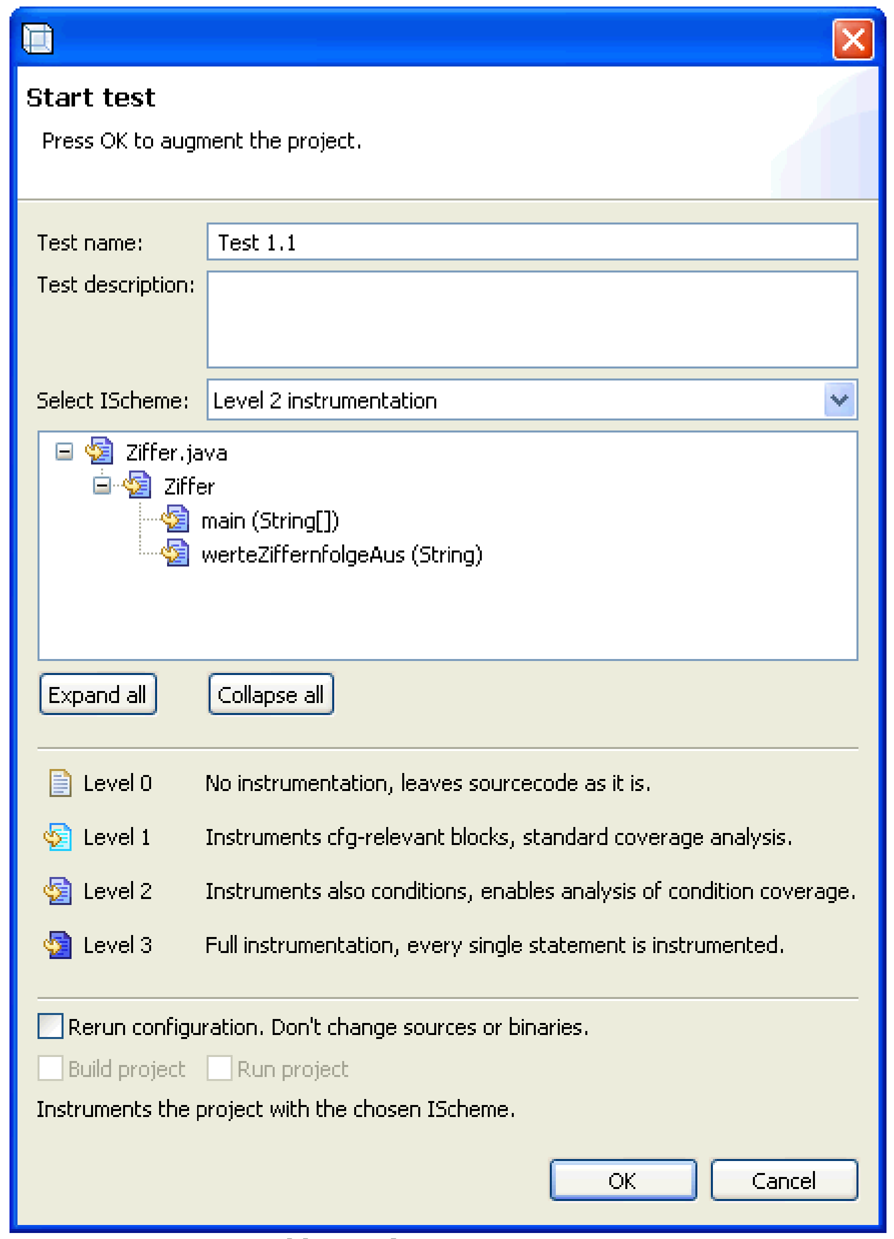


Abb.: Dialog *Start Test*

Nach dem Bestätigen des Dialoges wird die Datei *Ziffer.java* als *Ziffer.java.backup* gesichert und im Anschluss mit Instrumentierungsanweisungen versehen. Damit ist die Instrumentierungsvorbereitung in SOTA beendet und die Testphase kann beginnen.

4. Eclipse: Testphase - Kompilation

In Eclipse muss als erstes der geänderte Quellcode für das gesamte Projekt geladen werden. Dazu wählt man das Projekt *Ziffer* in der Projektübersicht aus und aktualisiert es über "F5" oder per Kontextmenü -> *Refresh*. Eclipse kompiliert daraufhin automatisch die neuen Quelldateien.

Die instrumentierten Quellen benötigen als Bibliothek die ASCLogger.jar, die unter Schritt 2. eingefügt wurde. Ohne die korrekte Einbindung der Bibliothek kommt es zu Fehlermeldungen.

5. Eclipse: Testphase - Programmtest

Nach erfolgreicher Kompilation ist das Programm zum Test bereit. Es wird in Eclipse über den

Button

Run As... gestartet. Beim ersten Start fragt Eclipse über eine Dialogbox danach, ob das Programm als Application oder Applet zu starten ist. Hier wählt man *Application*. Der darauffolgende Dialog fragt nach der zu startenden Application, hier ist *Ziffer* zu wählen, wonach das Programm startet. Für die nächsten Starts sollte Eclipse die einmal gewählte Startkonfiguration nehmen, es reicht dann, den Button
, welcher jetzt *Run Ziffer* heißt, zu drücken.

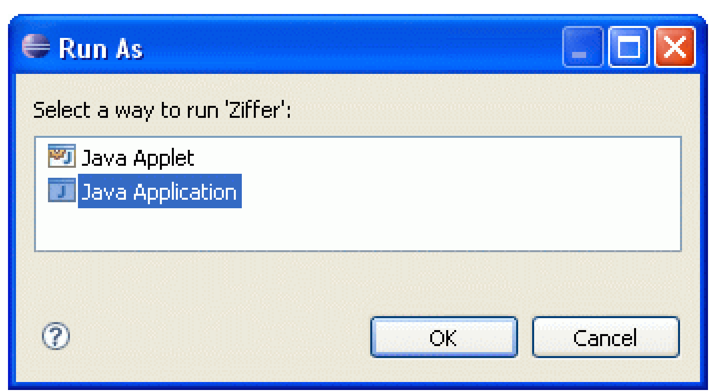


Abb.: Dialog *Start Test*

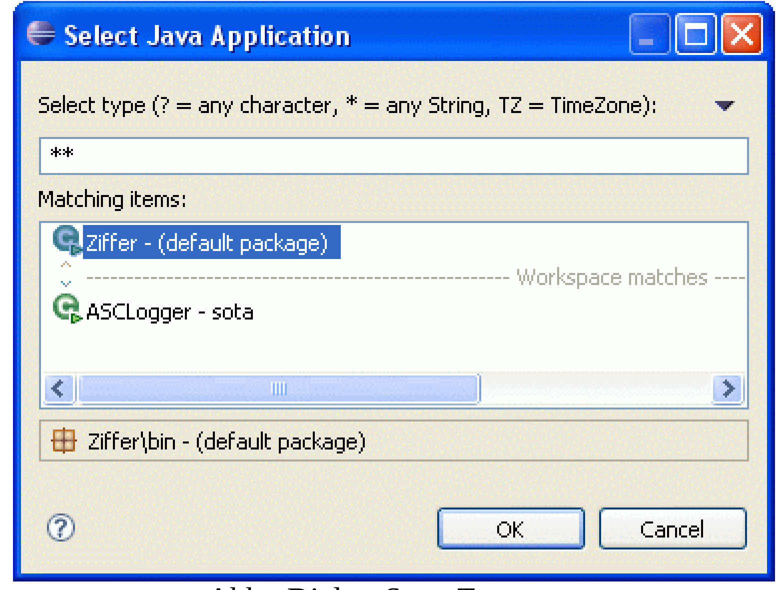


Abb.: Dialog *Start Test*

Die Parameter für den Programmstart unter Eclipse ließen sich in der Startkonfiguration des Projektes eintragen, aber da dieses Vorgehen für den einfachen Test etwas umständlich ist, wird empfohlen, den String in der Quellcodezeile

```
System.out.println(werteZiffernfolgeAus("."));
```

zu ändern und dann das Programm parameterlos zu starten. Die Console in Eclipse sollte nun die erfolgreiche Initialisierungsausschrift des ASCLoggers anzeigen, sowie dann das Ergebnis des Auswertungsversuches des Strings.

Zusätzlich sollte nach dem ersten Programmtest mit instrumentierten Quellen in der Projektübersicht von Eclipse die entsprechende Logdatei mit dem Names des Tests auftauchten. Wiederholte Tests führen dazu, dass die neuen Logdaten an diese Datei angehängt werden.

6. SOTA: Auswertungsphase - Rekonstruktion

Sind die Tests in Eclipse beendet, so muss dies SOTA über den Menüpunkt

Stop Test mitgeteilt werden. Die Quellen werden daraufhin wieder in den Originialzustand überführt. Alternativ dazu ließen sich die Quellen auch über den Menüpunkt
Restore Sources wiederherstellen.

7. SOTA: Auswertungsphase - Testauswertung

Um die Logdateien einzulesen, öffnet sich sofort nach Wahl des Menüpunktes
Stop Test ein Dialog, der alle Dateien mit der Endung log des Ausführungsverzeichnis für die Importierung auflistet. Die dort gewählten Logdateien werden gelesen, für die weitere Auswertung analysiert und erscheinen anschließend in der View *TestLogs*. Die dort markierten TestLogs werden nun zur Berechnung der Überdeckungsmaße herangezogen und bestimmen die Darstellung der Überdeckung in den Views *Source*, *CFG* und *Coverage*.

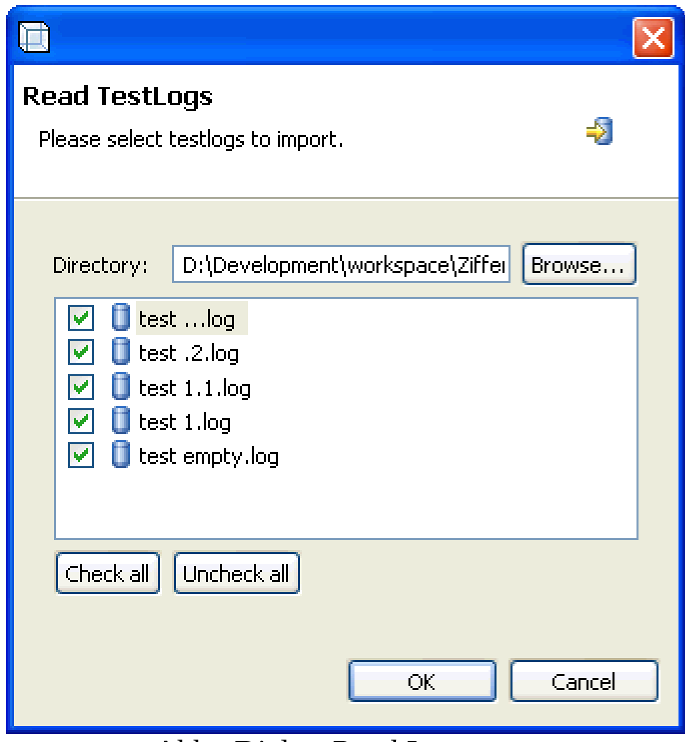


Abb.: Dialog *Read Logs*

Möchte man die Logdateien einlesen, ohne dass sich SOTA im Testmodus befindet, so ist der Menüpunkt

Read Logs zu wählen, der denselben Dialog öffnet.

8. SOTA: Auswertungsphase - Reporterstellung

Der Programmtest wird durch die Erstellung eines Reports abgeschlossen. Hierzu wählt man den Menüpunkt

Create Report, welcher abhängig von den Einstellung in den Präferenzen einen Dialog zur Eingabe des Dateinamens öffnet, oder aber einen generischen Namen wählt und die Reportdatei im Basisverzeichnis des Testprogrammes erstellt.

6.1.3 SOTA mit Ant-Buildfile und Startskript

Allgemeiner Ablauf

SOTA bietet die Möglichkeit, den Manuellen Programmtest durch die Verwendung zweier Skripte aus SOTA heraus zu vollziehen. Der Ablauf entspricht prinzipiell dem Ablauf aus Punkt 6.1.2, jedoch ist die dritte Phase, die Testphase unabhängig von Eclipse realisiert, welches lediglich zur Projekterstellung benötigt wird. Das folgende Datenflussdiagramm zeigt einen Überblick über die Arbeitsschritte in SOTA und an welchen Stellen die Skripte den Ablauf ergänzen.

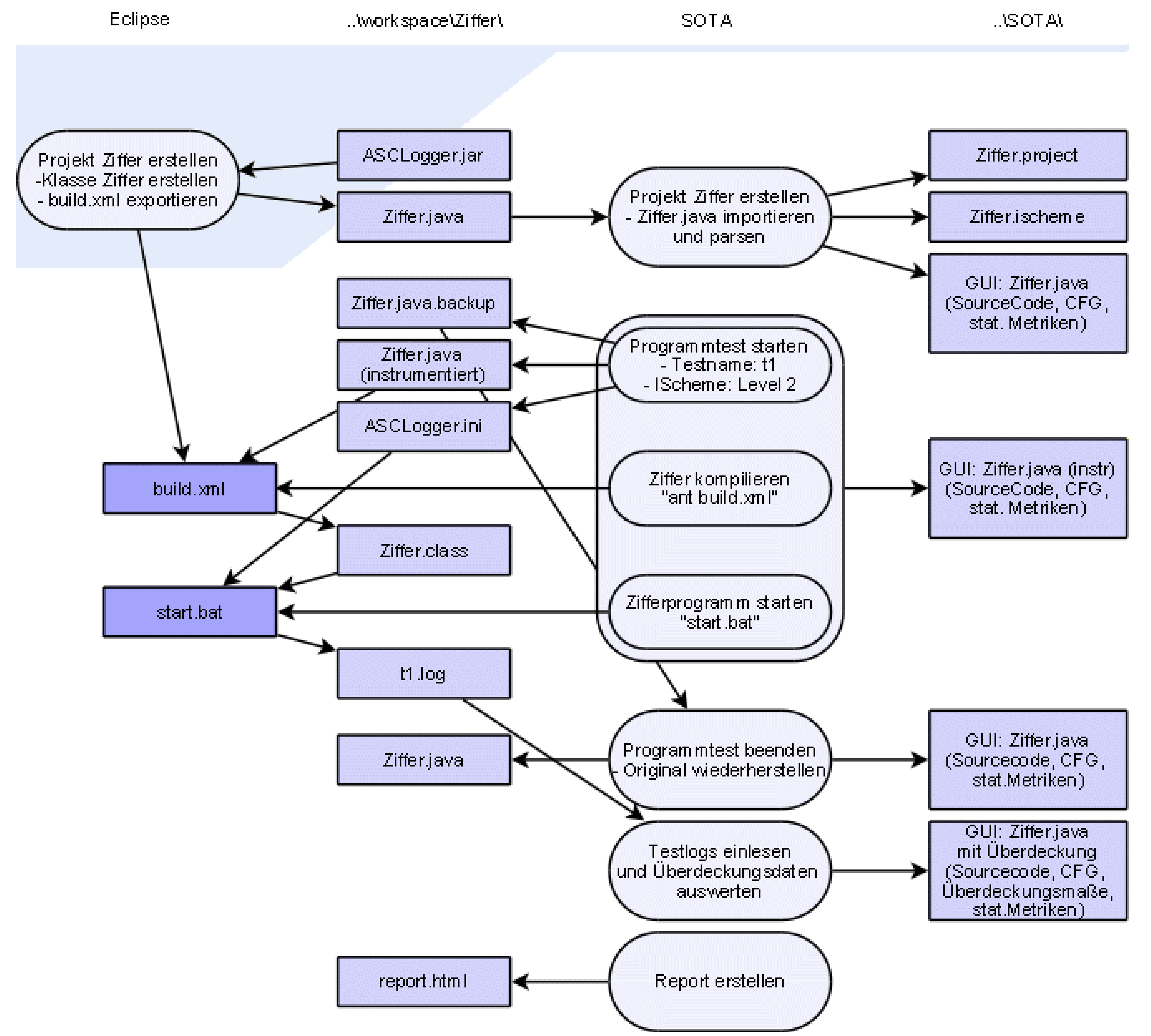


Abb.: DFD-Diagramm *Manueller Test mit Skripten*

Detailablauf

Der Detailablauf entspricht dem aus Punkt 6.1.2, es werden lediglich abweichende Schritte aufgeführt.

1. Programmerstellung

Zusätzlich zur Projekterstellung aus 6.1.2 wird in Eclipse ein xml-Buildfile exportiert. Dies erreicht man, indem man über das Kontextmenü des Projektes

Export ... aufruft und im sich öffnenden Dialog *General* -> *Ant Buildfiles* wählt.

Im zweiten Dialogfenser ist nur noch das entsprechende Projekt (hier: Ziffer) auszuwählen und nach dem Beenden wird eine XML-Datei namens "build.xml" im Basisverzeichnis des Projektes angelegt, die die Kompilation des gesamten Projektes ermöglicht.

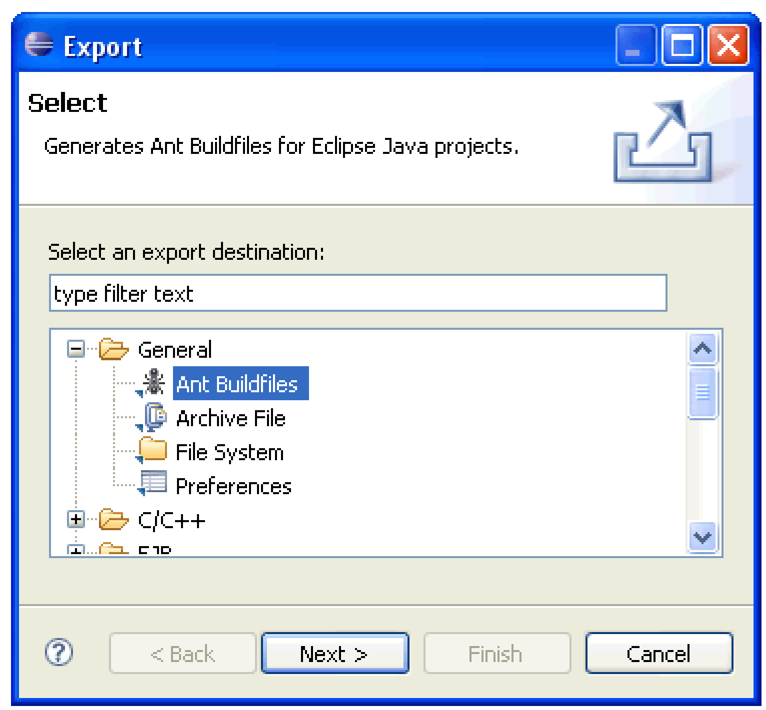


Abb.: Eclipse Export Dialog

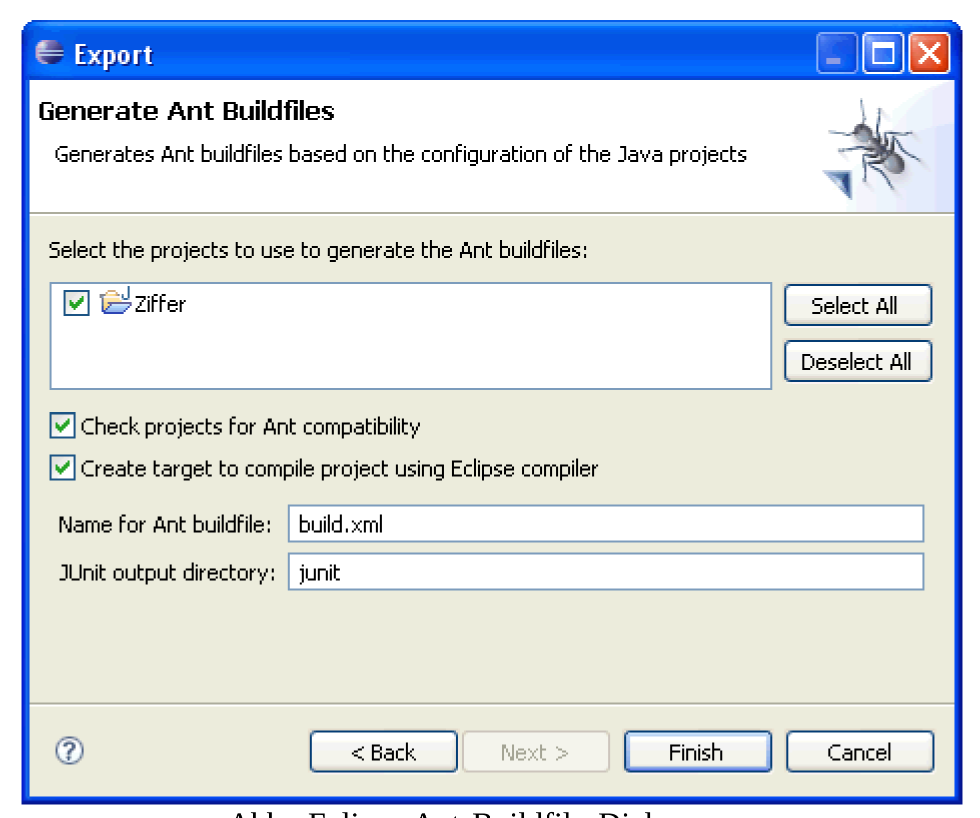


Abb.: Eclipse Ant-Buildfile-Dialog

Schließlich ist noch eine Batchdatei "Ziffer.bat" im Basisverzeichnis vom Ziffernprojekt anzulegen, mit welcher das Programm gestartet werden soll. Dazu ist einfach das Java-Kommando samt classpath dort einzutragen:

```
java -cp bin;lib/ASCLogger.jar; Ziffer
```

2. Vorbereitungsphase: Projekterstellung

Die Projekterstellung in SOTA funktioniert analog zu 6.1.2, mit

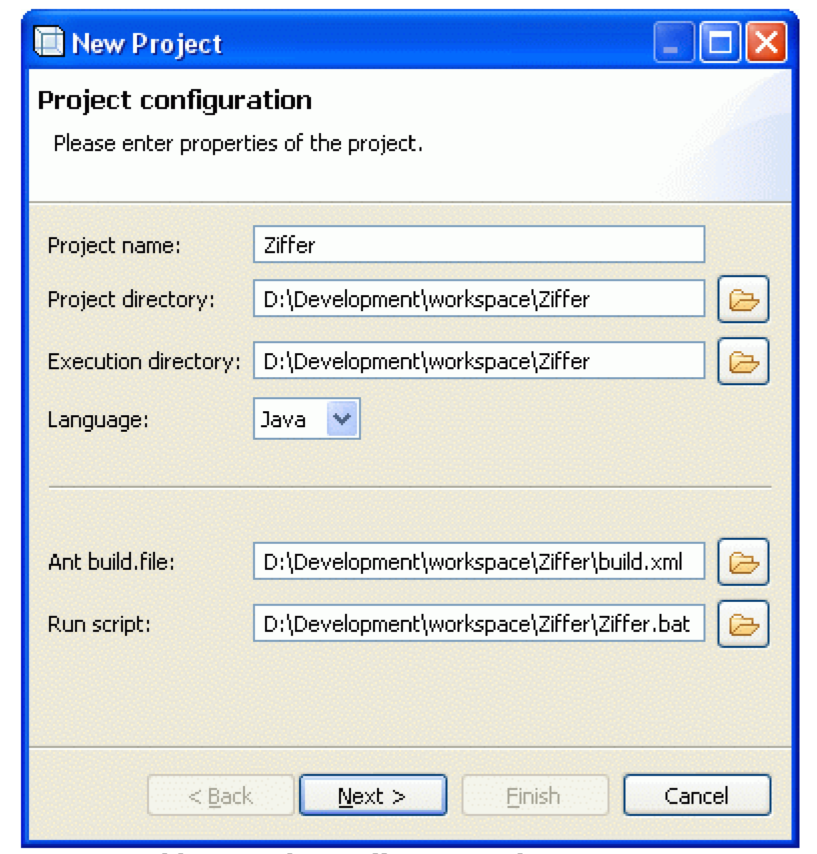


Abb.: Projekterstellung mit Skripten

dem einzigen Unterschied, dass nun die hier im ersten Schritt erzeugten Dateien "build.xml" und "Ziffer.bat" auf der ersten Wizardseite importiert werden. Diese Angaben können jedoch auch später noch über den Menüpunkt *Configure Project* geändert werden.

Damit die Kompilation anhand des Ant-Buildfiles gelingen soll, muss auch noch einmal für SOTA die ausführbare Ant-Datei "ant.bat" in den Präferenzen unter *Preferences -gt; General -gt; Location of Ant* eingebunden werden. Da die Einstellungen der Präferenzes projektübergreifend für SOTA gelten, ist dies nur einmal zu einzutragen.

3. Vorbereitungsphase: Instrumentierung / 4. Testphase: Kompilation / 5. Testphase: Programmtest

Durch die Einbindung beider Dateien ist der Manuelle Programmtest aus SOTA heraus möglich. Im unteren Bereich des Dialogs *Start Test* ist nun die Option *Build project* auswählbar, und wenn diese markiert ist, auch die Option *Run project*. Die erste Option führt nun dazu, dass nach dem Instrumentieren der Quellen nach dem ausgewählten IScheme das erstellte Ant-Buildfile abgearbeitet und so die instrumentierten Quellen kompiliert werden. Mit ausgewählter zweiter Option wird das angegebene Startskript ausgeführt und somit das Testprogramm gestartet. Die Startoption lässt in der gegenwärtigen Version von SOTA jedoch keine Parameterübergaben an das Testprogramm zu, wodurch der Einsatz für das Ziffernprogramm stark beschränkt wird. Um verschiedene Strings zu testen, wäre daher die Änderung des Startskripts vonnöten.

Start test

Press OK to augment the project.

Test name: test .

Test description:

Select IScheme: Level 2 instrumentation

Ziffer.java
Ziffer
main (String[])
werteZiffernfolgeAus (String)

Expand all | Collapse all

Level 0 No instrumentation, leaves sourcecode as it is.
Level 1 Instruments cfg-relevant blocks, standard coverage analysis.
Level 2 Instruments also conditions, enables analysis of condition coverage.
Level 3 Full instrumentation, every single statement is instrumented.

Rerun configuration. Don't change sources or binaries.
Build project Run project
Instruments the project with the chosen IScheme, compiles and runs project with Ant.

OK Cancel

Abb.: Dialog *Start Test* mit Kompilations- und Startoption

6. Auswertungsphase: Rekonstruktion / 7. Auswertungsphase: Testauswertung / 8. Auswertungsphase: Reporterstellung

Die restlichen Schritte des Manuellen Programmtest verlaufen ohne Änderung wie in 6.1.2.

6.2 Test mit einem externem Testsystem (ATOSj) - HU-Seminarorganisation

6.2.1 ATOSj und HUSemOrg

Voraussetzung für die Nutzung von ATOSj als externes Testsystem und von HUSemOrg als Testprogramm ist ihre Installation.

Anleitungen hierzu findet man unter:

- Installationshinweise für das Seminarorganisations-Programm HUSemOrg:
- Installationshinweise für ATOSj:

- ATOSj: Einrichten eines Projektes für die Seminarorganisation:

6.2.2 SOTA und ATOSj

Allgemeiner Ablauf

Der Testablauf orientiert sich am Manuellen Test, es folgt allerdings die Auslagerung des Testschrittes in das externe Testprogramm. Damit ergeben sich folgende Phasen im Testbetrieb:

1. Eclipse: Programmeingabe
2. SOTA: Vorbereitungsphase
3. ATOSj: Testphase
4. SOTA: Auswertungsphase.

Auch hier wird Eclipse durch die Verwendung eines Ant-Buildfiles lediglich zur Programmerstellung genutzt und spielt im weiteren Testverlauf keinerlei Rolle mehr. Das folgende Datenflussdiagramm gibt einen Überblick, an welcher Stelle das externe Testsystem in den Testphasen auftritt.

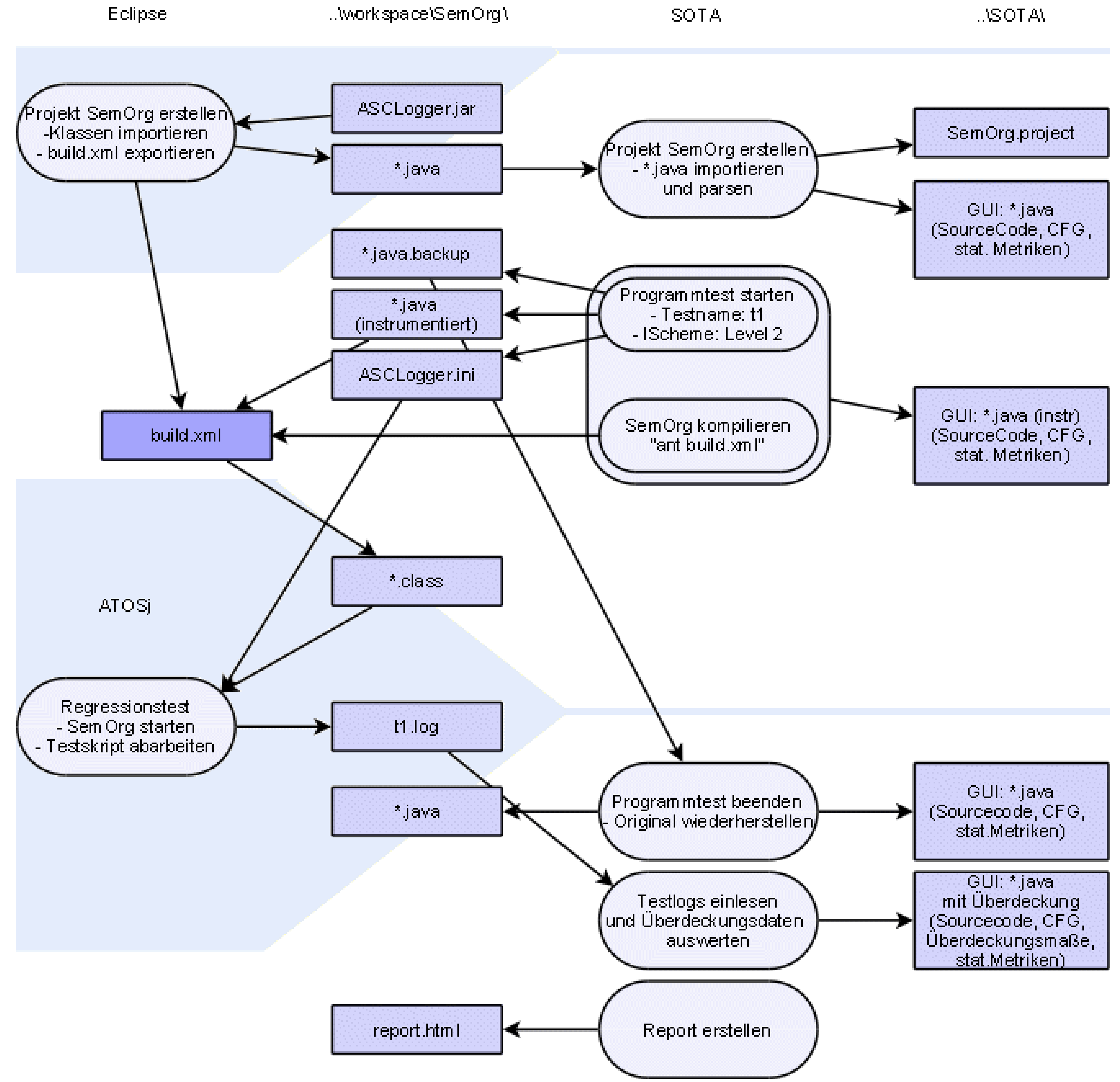


Abb.: DFD-Diagramm *Manueller Test mit Skripten*

Detailablauf

Der Detailablauf entspricht bis auf wenige Ausnahmen, die im Folgenden kurz ausgeführt werden, dem des Manuellen Tests.

1. Programmerstellung

Das Programm HuSemOrg wird in den Ordner *workspace* von Eclipse entpackt und enthält dabei schon die nötigen Anpassungen für die Verwendung mit SOTA. Analog zu 6.1.2 ist nun in Eclipse ein Projekt *husemorg* zu erstellen und die Bibliothek *ASCLogger.jar* zum Buildpath hinzuzufügen.

2. Vorbereitungsphase: Projekterstellung / 3. Vorbereitungsphase: Instrumentierung / 4. Testphase: Kompilation

Die nächsten drei Schritte beziehen sich nur auf die Vorbereitung in SOTA und sind identisch mit dem Manuellen Test.

5. Testphase: Programmtest

Nachdem nun die instrumentierten Quelldateien kompiliert wurden, wird für den Programmtest selbst ATOSj gestartet und auf den instrumentierten Klassen die Regressionstests ausgeführt. SOTA kann dafür beendet werden. Die Benutzung von ATOSj ändert sich in keiner Weise gegenüber dem normalen Regressionstest ohne SOTA, es gibt keine Interaktion zwischen den beiden Programmen.

6. Auswertungsphase: Rekonstruktion / 7. Auswertungsphase: Testauswertung / 8. Auswertungsphase: Reporterstellung

Die restlichen Schritte verlaufen ohne Änderung wie in 6.1.2.

6.3 Automatisches Testsstem - SOTA-ATM

SOTA-ATM (Automatisches Test-Modul) ist eine Bibliothek, die die Testfunktionalität von SOTA kapselt und völlig ohne GUI auskommt. Dadurch wird ermöglicht, die Instrumentierung von Projekten sowie die Auswertung der Logdateien aus anderen Programmen heraus automatisch zu steuern.

Zwei Steuerungsansätze werden für das Modul angeboten. Zum einen das Starten von SOTA-ATM als ausführbares Jar, welches über Kommandozeilenparameter gestartet und gesteuert werden kann und somit einfach in Skripten einsetzbar ist. Und zum anderen als integrierbare Bibliothek, welche Schnittstellen für die Funktionen zum Testen des Projektes bietet, die aus einem anderen Programm heraus aufgerufen werden können.

6.3.1 SOTA-ATM über Kommandozeilenaufruf

Parameter - Überblick

SOTA-ATM ist ein ausführbares Jar und lässt sich über die Kommandozeile mit verschiedenen Parametern starten. Die notwendigen Projektinformationen erhält SOTA-ATM entweder durch das Einlesen einer Projektdatei (-*p*) oder indem die notwendigen Werte beim Programmaufruf übergeben werden (-*n*). Eine Projektdatei kann über die graphische Benutzeroberfläche von SOTA, dem Nutzen der Option -*n* von SOTA-ATM oder auch manuell erstellt und mit Werten gefüllt werden.

Die weiteren Optionen verursachen die Ausführung der verschiedenen Teilfunktionen des Moduls im Test. Die wesentlichen Aspekte sind hier das Instrumentieren der Quelldateien (-*i*), das

Wiederherstellen der originalen Quelldateien (*-z*), sowie die Auswertung von Tests durch das Einlesen der entsprechenden Logdateien (*-t*) und dem anschliessenden Erstellen eines Reports (*-r*).

Zusätzliche Funktionen, die damit kombiniert werden können, sind das Kompilieren (*-c*) und Starten des Testprogrammes (*-s*), falls die entsprechenden dazu notwendigen Dateien vorliegen. Die Reihenfolge der Optionen ist ohne Bedeutung.

```
Usage: java -jar SOTA-ATM.jar [-options]

options include:
        -c [ <ant-buildfile> ]
                compile sourefiles; only if ant-buildfile is provided
        -i ( Level1 | Level2 | Level3 | <ischeme-name> )
                instrument sourcefiles according to chosen level or IScheme
        -n <name> <lang> <project-dir> [ <exec-dir> <src dir> ]
                create new project file
        -p <name>.project
                open the project file
        -r [ <report-file> ]
                create report-file; if no name is provided it will be stored in
"report.html"
        -s [ <runscript> ]
                start project; only if runscript is provided
        -t <testname>.log [ <testname>.log ... ]
                name of testlog to create or to import
        -z      restore original sources
```

Parameter - detailliert

-c [<ant-buildfile>]

Der Parameter *c* führt dazu, dass das Projekt im Anschluss an die Dateioperationen kompiliert wird. Dazu ist es notwendig, dass in der Projektdatei ein Verweis auf Apache Ant eingetragen ist, und ein Ant-Buildfile entweder direkt nach *-c* übergeben wird, oder in der Projektdatei eingetragen ist.

-i (Level1 | Level2 | Level3 | <ischeme-name>)

Mit dem Parameter *i* werden alle Quellen des Projektes als Backup gespeichert und danach instrumentiert. Es muss entweder der Name eines ISchemes aus der Projektdatei folgen, oder einer der Werte: "Level1", "Level2", "Level3", die für eine vollständige Instrumentierung des gesamten Projektes nach dem entsprechenden Level stehen. Das Projekt wird nur instrumentiert, wenn keine der Quelldateien schon instrumentiert ist.

Der Parameter *i* verlangt die Angabe eines Testnamens über den Parameter *t* und schließt die Verwendung der Parameters *r* zur Reporterzeugung und *z* zur Wiederherstellung der Quellen aus.

-n <name> <lang> <project-dir> [<exec-dir> <src dir>]

Der Parameter *n* führt zu einer Neuerstellung eines Projektes anhand der übergebenen Werte. Mindestens erforderlich ist die Angabe des Projektnamens, der verwendeten Programmiersprache und dem Projektverzeichnis des Testprogrammes. Die optionalen Parameter umfassen das Ausführungsverzeichnis und das Quellverzeichnis. Fehlen die optionalen Parameter, fällt für SOTA-ATM das Projektverzeichnis mit dem Ausführungsverzeichnis und dem Verzeichnis der Quelldateien zusammen.

Nach der Projekterstellung wird die Projektdatei unter dem Namen <name>.project gesichert. Der Parameter *n* schließt die Verwendung des Parameters *p* aus.

-p <name>.project

Mit diesem Parameter wird eine Projektdatei übergeben werden, welche die Informationen zur Charakterisierung des Projektes enthält. Solch eine Projektdatei kann zum einen über die graphische Benutzeroberfläche von SOTA, durch die Projektneuerstellung in SOTA-ATM über den Parameter *-n*

oder manuell erstellt werden. Der Parameter *p* schließt die Verwendung des Parameters *n* aus.

-r [<report-file>]

Für die Testlogdateien, die über den Parameter *t* eingelesen wurden, werden die berechneten Überdeckungsmaße für das Projekt mit dem Parameter *r* in eine Reportdatei geschrieben. Folgt dem Parameter eine html-Datei, so wird in diese der Report geschrieben. Andernfalls erfolgt dies in die Datei "report.html".

Der Parameter *r* benötigt die Angabe mindestens einer Logdatei durch *t* und schließt die Verwendung des Parameters *i* für die Instrumentierung aus. Sind die Quelldateien instrumentiert, werden sie vor dem Testeinlesen wieder in den originalen Zustand überführt.

-s [<runscript>]

Mit dem Parameter *s* kann das Starten des Testprogrammes veranlasst werden. Dazu ist entweder die Angabe eines Runscriptes nach dem Parameter oder in der Projektdatei erfolderlich.

-t <log-file>.log [<log-file>.log ...]

Dieser Parameter hat zweierlei Bedeutung. Beim Instrumentieren mittels *i* legt er den Namen des Tests fest und damit auch, wie die zu erstellende Logdatei heißt. Für das Erstellen des Reports durch den Parameter *r* werden hier alle Testlogdateien aufgelistet, die importiert und für die Überdeckungsberechnung ausgewertet werden sollen.

-z

Der Parameter *z* führt dazu, dass alle Quelldateien des Projektes vom Backup wiederhergestellt werden. Er kann nicht zusammen mit dem Parameter *i* zur Instrumentierung verwendet werden.

Beispielnutzung

java -jar SOTA-ATM.jar -n Ziffer Java /workspace/Ziffer

Es wird ein Projekt "Ziffer" der Programmiersprache Java erstellt, dessen Projektverzeichnis unter "/workspace/Ziffer" zu finden ist. Von dort werden auch alle java-Dateien in das Projekt importiert. Nach der erfolgreichen Projekterstellung werden die Projektinformationen in die Datei "Ziffer.project" gespeichert. Dort können auch verschiedene Einstellung des Projektes, wie z.B. zu verwendende Buildfiles, verändert werden.

java -jar SOTA-ATM.jar -p Ziffer.project -i Level2 -t test1 -c -s

Dieser Aufruf des Moduls führt dazu, dass das Ziffer-Projekt geparst, vollständig nach Level 2 instrumentiert und im Anschluss kompiliert und gestartet wird. Außerdem wird als Name für den Test "test1" übergeben. Die Reihenfolge der Optionen ist ohne Bedeutung.

java -jar SOTA-ATM.jar -p Ziffer.project -z -c

Mit diesem Aufruf stellt man den Originalstand der Quellen und Binaries wieder her.

java -jar SOTA-ATM.jar -p Ziffer.project -t test1 -r

Dieser Aufruf des Moduls führt dazu, dass für das Projekt Ziffer der Test "test1" eingelesen wird und die dabei berechneten Überdeckungsdaten in die Standardreportdatei "report.html" geschrieben werden. Sind die Quelldateien instrumentiert, werden sie zu Beginn in den originalen Zustand

überführt.

6.2.2 SOTA-ATM API

Überblick

Bindet man SOTA-ATM als Bibliothek in sein Programm ein, wird damit die Non-GUI-Funktionalität von SOTA für die statische Analyse und den Überdeckungstest zur Verfügung. Die Javadoc-Dokumentation ist hier zu finden.

Zentrale Klasse für den Programmtest ist die Klasse SotaATM, deren Objekte jeweils eine Testinstanz von SOTA darstellen. Konfiguriert wird eine Testinstanz entweder über das Laden einer Projektdatei oder der Übergabe einer ProjectConfiguration-Instanz, die analog zu einer Projektdatei alle relevanten Informationen für den Programmtest enthält. An dieser Testinstanz kann die Funktionalität von der Instrumentierung bis zur Reporterstellung genutzt werden (siehe javadoc).

Direkt nach dem Öffnen des Projektes lässt sich ein Metrics-Objekt von der Testinstanz zurückgeben, welche alle Maße der statischen Analyse enthält. Die Überdeckungsmaße sind im selben Objekt nach dem Programmtest und dem Auswerten der erzeugten Logdateien abrufbar.

Zur Konfiguration der Instrumentierung für einen Test, lassen sich variable ISchemes definieren, durch welche jeder einzelnen Struktur (Datei, Klasse, Funktion) des Testprojektes einem Instrumentationslevel zugewiesen werden kann. Die gesamte Instrumentierung des Projektes nach einem Instrumentationslevel lässt sich durch die Verwendung eines GlobalIScheme erreichen.

Beispielimplementation des manuellen Tests von HuSemOrg

Der folgende Java-Programmcode liest die vorgegebene Projektdatei ein, startet einen Test nach Instrumentationslevel 2, kompiliert und startet das Testprogramm. Nach Beendigung des Tests werden die originalen Quelldateien wiederhergestellt, das Projekt neu kompiliert und aus dem importierten Test ein Html-Report erstellt.

```
String projectFile = "D:/Development/eclipse/husemorg.project";
String reportFile = "report_test_1.html";
String testName = "husemorg_test_1";
String testDesc = "Test 08/15";

TreeSet<String> testSet = new TreeSet<String>();
testSet.add(testName);

SotaATM atm = new SotaATM(fileName);
atm.startTest(testName, testDesc, new GlobalIScheme("Level2", 2) , true);
atm.stopTest(testSet, true);
atm.createReport(reportFile);
```

SOTA Benutzer-Dokumentation - Version 1.0

7 Anhang

7.1 Statische Maße

Unter statische Maße subsummiert SOTA alle Maßzahlen des Projektes, die durch statische Analyse des Quellcodes durch SOTA gewonnen werden. Sie werden beim Parsen des Quellcodes ermittelt und bedürfen im Gegensatz zu den Überdeckungsmaßen keiner Ausführung des Programmes. Die Maße erlauben zum einen die Abschätzung der Komplexität des Quellcodes in Hinblick auf verschiedene Kriterien und ermöglichen damit auch eine Verbesserung der Struktur des Quellcodes. Zum anderen liefern sie eine Abschätzung des nötigen Testaufwandes bzw. der Anzahl an verschiedenen Tests für die einzelnen Kriterien.

Die statischen Maße sind direkt nach dem Laden des Projektes in der View*Metrics* für alle Strukturen des Projektes einsehbar. Für die zyklomatische und essentielle Komplexität entspricht dieser Wert bei Klassen, Dateien und dem Projekt dem Maximum der Werte ihrer untergeordneten Funktionen, für alle anderen Maße werden die Werte aufsummiert.

Anmerkungen zu den ModBI- und BI-Werten: Der gesamte Komplex Ausnahmebehandlung macht eine exakte Bestimmung von Pfaden unmöglich. Der berechnete Wert ist daher immer eine untere Grenze, d.h. die minimale Anzahl an Teilpfaden beim ModBI-Test und die minimale Anzahl an Pfaden beim BI-Test, die beim Test zu erreichen möglich ist.

7.1.1 Zyklomatische Komplexität

Die zyklomatische Komplexität einer Funktion ist anhand ihres Kontrollflussgraphen definiert, der die möglichen Pfade des Programmablaufes und alle seine Verzweigungen darstellt (vgl. View *CFG*).
Sind e und n die Anzahl der Kanten sowie der Knoten des Kontrollflussgraphen G, so ist die zyklomatische Zahl $z(G)$ definiert als: $z(G) = e - n + 2$.
Eine Funktion ohne Verzweigungen im Programmfluss hat demnach immer eine zyklomatische Komplexität von 1, jede Verzweigung im Programmablauf, z.B. durch eine if-Anweisung, erhöht die zyklomatische Komplexität um 1.

7.1.2 Essentielle Komplexität

Die Definition der essentiellen Komplexität schließt an die der zyklomatischen Komplexität an.
Wenn man von einem gegebenen Kontrollflussgraphen einer Funktion rekursiv alle primitiven Kontrollstrukturen entfernt, solange dies möglich ist, dann wird die zyklomatische Komplexität des entstandenen Graphen G' definiert als essentielle Komplexität des Graphen G: $e(G) = z(G')$.
Dabei entsprechen die primitiven Kontrollstrukturen allen einfachen Kontrollstrukturen, die außer

den break-Anweisungen bei der Switch-Anweisung keine Sprünge aufweisen. Das Vorhandensein von Sprüngen aus einer Kontrollstruktur heraus macht diese und alle sie einschließenden Strukturen nichtreduzierbar und trägt somit zur Steigerung der essentiellen Komplexität bei.

7.1.3 Lines of code (LOC)

Als eines der primitivsten Maßzahlen zum Quellcode wird hier die Anzahl der Quellcodezeilen aufgeführt, die die entsprechende Struktur umfasst. Im Gegensatz zu allen anderen Maßzahlen, die SOTA berechnet, ist LOC stark von der Quellcodestrukturierung und auch von der Art der Kommentierung abhängig und daher vorsichtig zu bewerten.

7.1.4 Anzahl der Anweisungen (#Statements)

Im Gegensatz zur Lines-of-code-Maßzahl bietet die Anzahl der Anweisungen eine objektive, von der Quellcodeformatierung unabhängige Maßzahl für den Umfang des Projektes. Hierzu werden alle ausführbaren Anweisungen für alle Strukturen aufsummiert. Der Anweisungsüberdeckungstest berechnet sich durch den Vergleich der ausgeführten Anweisungen mit der Anzahl aller Anweisungen.

7.1.5 Anzahl der Abzweigungen (#Branches)

Die Anzahl der Abzweigungen ist in SOTA funktionell als Mittel zur Berechnung der Zweigüberdeckung definiert. Während die Anzahl der Verzweigungen in einer Funktion der zyklomatischen Komplexität - 1 entspricht, wird hier die Anzahl der Abzweigungen als Summe der Ausgänge aller verzweigenden Knoten definiert. Für eine Funktion ohne Verzweigungen ist also die Anzahl der Verzweigungen null, für jede hinzugefügte if-Anweisung erhöht sich der Wert um zwei.

7.1.6 Anzahl der Modifizierten Boundary-Interior-Pfade (#ModBI)

Die Anzahl der Modifizierten Boundary-Interior-Pfade entspricht der Anzahl an Teilpfaden durch den Kontrollflussgraphen, die für die vollständige Erfüllung des Modifizierten Boundary-Interior-Pfadüberdeckungstest getestet werden müssen. Die verschiedenen Arten von Teilpfaden sind dabei nach Liggesmeyer (*Software-Qualität*, 2002) wie folgt definiert:

- alle ausführbaren Pfade durch eine Funktion, die abweisende Schleifen nicht betreten und nicht-abweisende Schleifen nicht wiederholen,
- alle ausführbaren Teilpfade einer jeden Schleife, die den Schleifenrumpf genau einmal ausführen, wobei das Verhalten in Bezug auf umschlossene Schleifen nicht beachtet wird,
- alle ausführbaren Teilpfade einer jeden Schleife, die den Schleifenrumpf mindestens zweimal ausführen, wobei das Verhalten in Bezug auf umschlossene Schleifen und etwaige anschließende Durchläufe des Schleifenrumpfes nicht beachtet wird.

In der View *CFG* kann man für jede einzelne Schleife die Anzahl der für sie zu testenden Teilpfade nach dieser Definition aus der Knoteninfo (durch Doppelklick auf den entsprechenden Knoten) erfahren. Hier wird dieser Wert unter "#ModBI" aufgeführt. In der Knoteninfo des Funktionsknoten findet man sowohl den Wert für die gesamte Funktion, als auch die Anteile durch die Teilpfade der Schleifen sowie der Teilpfade durch die gesamte Funktion.

7.1.7 Anzahl der Boundary-Interior-Pfade (#BI)

Analog zum vorhergehenden Wert wird hier die Anzahl der ausführbaren Boundary-Interior-Pfade für jede Funktion angegeben, bzw. für Klassen, Dateien und das Projekt die Summe aller sie enthaltenen Werte. Die entsprechenden Pfade sind definiert als alle ausführbaren Pfade durch die Funktion, wobei zur Begrenzung der Pfadanzahl gilt, dass bei Vorkommen von Schleifen lediglich jene Pfade getestet werden müssen, die für jede Schleife

- die Schleife überspringen, d.h. den Schleifenkörper nicht ausführen (dies ist bei do-while-Schleifen nicht möglich),
- den Schleifenkörper genau einmal durchlaufen,

- den Schleifenkörper mind. zweimal durchlaufen, wobei nur die ersten beiden Iterationen betrachtet werden.

7.1.8 Anzahl der Anweisungen mit logischen Bedingungen (#ConditionStmts.)

Zur Berechnung der Anzahl der Anweisungen mit logischen Bedingungen werden alle Vorkommen von Anweisungen mit auswertbaren logischen Bedingungen im Quellcode aufsummiert. Explizit nicht gezählt werden Endlosschleifen ("while(true)") und Schleifen, welche über eine Menge iterieren ("for(Item item : set)").

7.1.9 Anzahl der logischen Atome (#Atoms)

Diese Maßzahl entspricht der Summe der auswertbaren atomaren Bedingungen aus allen logischen Bedingungen. Die logischen Atome *true* und *false* werden hierbei nicht mitgezählt, da sie nicht auswertbar in Bezug auf den Überdeckungstest für Bedingungen sind und keinen Einfluss auf den Kontrollfluss nehmen.

7.1.10 Anzahl der logischen Bedingungen (#Conditions)

Die Anzahl der logischen Bedingungen enthält die Summe aller atomaren und zusammengesetzten Bedingungen. Dieser Wert ist für die Berechnung der minimal Mehrfach-Bedingungsüberdeckung wichtig.

7.2 Überdeckungsmaße

Der eigentliche Zweck SOTAs liegt in der Bewertung von Programmtests durch die Berechung von Überdeckungsmaßen. Durch das Einfügen von Instrumentierungen wird während des Programmtests eine Logdatei mit den notwendigen Daten erstellt, die es SOTA ermöglichen, im nachhinein den Programmverlauf und die Auswertungen der Bedingungen zu rekonstruieren. Aus diesen Daten werden für die einzelnen Tests die gängigsten Überdeckungsmaßzahlen bestimmt und in der View *Coverage* aufgelistet.

7.2.1 Function-Entry-Exit-Coverage (FEEC)

Der Function-Entry-Exit-Coveragetest fordert für die vollständige Überdeckung, dass für jede Funktion alle Eingänge und alle Ausgänge genommen werden. Seine Erfüllung wird wie folgt berechnet:

- FEEC = (#besuchte Funktionseingänge + #besuchter Funktionsausgänge) / (#Funktionseingänge + #Funktionsausgänge)

Bei Java gibt es nur je einen Eingang für eine Funktion. Als mögliche Ausgänge wird das normale Funktionsende, falls es erreicht sein sollte, sowie alle return-Anweisungen und alle throw-Anweisungen außerhalb von try-Strukturen gezählt.

7.2.2 Anweisungsüberdeckung (C0)

Für die Anweisungsüberdeckung ist es notwendig, dass jede Anweisung im Quellcode ausgeführt wurde. Da nur für Quellcode nach Instrumentierungslevel 3 jede Anweisung bei der Ausführung in der Logdatei vermerkt wird, wird im Normalfall nach dem Programmtest die Überdeckung der Anweisungen aus den geloggten Eckdaten des Kontrollflusses propagiert.

- C0 = #überdeckte Anweisungen / #Anweisungen

Anmerkung: In der View *CFG* entsprechen nicht alle Knoten Anweisungen und nicht jede Anweisung entspricht einem Knoten. Es lässt sich deshalb nicht aus den überdeckten Knoten des Kontrollflussgraphen die C0-Überdeckung berechnen, die Grundlage bildet hier der Wert *#Statements* aus der View *Metrics*.

7.2.3 Zweigüberdeckung (C1)

Die vollständige Zweigüberdeckung wird erreicht, wenn alle Zweige des Kontrollflussgraphen überdeckt wurden. Die Berechnung der prozentualen Überdeckung wird in der Praxis unterschiedlich gehandhabt, der Einfachheit halber berechnet SOTA dies anhand der Abzweigungen (vgl. 7.1.5) wie folgt:

- C1 = #überdeckte Abzweigungen / #Abzweigungen

7.2.4 Einfache Bedingungsüberdeckung (C2)

Die einfache Bedingungsüberdeckung testet ausschließlich, ob alle logischen Atome der Bedingungen sowohl wahr als auch falsch ausgewertet wurden. Damit muss jedoch noch keine Zweigüberdeckung als Minimalziel erreicht worden sein, weswegen die einfache Bedingungsüberdeckung noch keinen großen Aussagewert besitzt. Für die Berechnung der prozentualen Überdeckung zählt SOTA alle Auswertungen der Atome und vergleicht diese mit dem Zielwert.

- C2 = (#wahr-Auswertungen aller Atome + #false-Auswertungen aller Atome) / 2 * #Atome

7.2.5 Minimal Mehrfach-Bedingungsüberdeckung (MMCC)

Als praxistaugliche Bedingungsüberdeckung, die auch die Zweigüberdeckung einschließt, hat sich die minimale Mehrfach-Bedingungsüberdeckung etabliert. Hierzu werden analog zur C2 alle Auswertungen nicht nur der logischen Atome, sondern auch der zusammengesetzten, komplexen Bedingungen betrachtet. Diese müssen während des Tests jeweils zu wahr und falsch ausgewertet werden. Die Anzahl der zu untersuchenden logischen Strukturen entspricht hier der unter 6.1.10 aufgeführten Anzahl der logischen Bedingungen.

- MMCC = (#wahr-Auswertungen aller Bedingungen + #false-Auswertungen aller Bedingungen) / 2 * #Bedingungen

7.2.6 Modifiziert Bedingungs-Entscheidungsüberdeckung (MCDC)

Ein noch schärferes Testkriterium liefert der modifizierte Bedingungs-Entscheidungsüberdeckungstest. Hierzu ist es nicht nur notwendig, dass alle logischen Atome einer jeden Bedingung die Werte wahr und falsch annehmen, sondern es muss auch für jedes einzelne Atom gelten, dass es Wertbelegungen für diese Bedingung gab, die sich ausschließlich in diesem Atom unterschieden und zu einer unterschiedlichen Auswertung der gesamten zusammengesetzten Bedingung führte. Damit wird sichergestellt, dass getestet wurde, dass das Ändern das Wahrheitswertes eines jeden Atoms auf die Gesamtbedingung einen Einfluss hat. Die beide Wahrheitsvektoren einer Bedingung, die diese Forderung für ein Atom erfüllen, werden MCDC-Paar genannt. Die Überdeckungsmaßzahl berechnet sich dann anhand der gefundenen MCDC-Paare wie folgt:

- MCDC - modifizierte Bedingungs/Entscheidungsüberdeckung = #aufgetretener MCDC-Paare / #Atome

7.2.7 Mehrfach-Bedingungsüberdeckung (C3)

Den umfangreichsten Bedingungstest fordert der Mehrfach-Bedingungsüberdeckungstest, da er den Test aller Wahrheitsvektoren einer jeden Bedingung fordert. Der Testaufwand steigt damit jedoch exponentiell mit der Anzahl der Bedingungen an. Hinzu kommt, dass in den meisten Fällen nicht alle Kombinationen von Wahrheitswerten der Atome überhaupt belegbar sind, sondern viele rein praktisch unmöglich sind, ohne dass dies einfach zu erkennen wäre. Der Testauwand von 2^(#Atome) wird lediglich durch die Anwendung von Short-circuit-Operatoren reduziert, welche die Auswertung der Bedingungen stoppt, sobald das Ergebnis der Gesamtbedingung unumstößlich fest steht.

- C3 = #ausgewertete Wahrheitsvektoren / #mögliche Wahrheitsvektoren

7.2.8 Modifizierte Boundary-Interior-Pfadüberdeckung (ModBI)

Der Modifizierte Boudary-Interior-Pfadüberdeckungstest ist ein von Liggesmeyer vorgeschlagener Test, der eine nochmalige Reduktion der nötigen Testfälle gegenüber dem Boundary-Interior-Pfadtest beinhaltet (siehe Definition in 7.2.9). Für die Berechnung der Überdeckungsmaßzahl werden für alle Pfade durch eine Funktion während des Programmtests die MBI-Pfade berechnet, die sie überdecken, und die Summe dieser überdeckten Teilpfade wird mit der Anzahl der möglichen MBI-Pfade, definiert in 7.2.9, verglichen.

Da die Anzahl der ModBI-Pfade nur ein Minimum möglicher Teilpfade nach diesem Kriterium ist, können in der Praxis mehr MBI-Pfade durchlaufen werden (z.B. durch Ausnahmen), als durch diese minimale Schranke vorgegeben wurde. In diesem Fall wird der Überdeckungswert natürlich auf 1 begrenzt.

- ModBI = #durchlaufener MBI-Pfade / #möglicher MBI-Pfade

7.2.9 Boundary-Interior-Pfadüberdeckung (BI)

Die Boundary-Interior-Pfadüberdeckung wird analog zur Modifizierten Boundary-Interior-Pfadüberdeckung berechnet, wobei lediglich für alle Pfade durch eine Funktion die BI-Pfade berechnet und deren Anzahl mit der möglichen Anzahl an möglichen BI-Pfaden verglichen wird.

- BI = #durchlaufener BI-Pfade / #möglicher BI-Pfade

7.3 Instrumentationslevel

Um dem Nutzer zu erlauben, den mitunter sehr hoch ausfallenden Speicherbedarf der Logdateien sinnvoll und variabel zu begrenzen, lässt sich der Quellcode in verschiedenen Stufen instrumentieren. Eine Konfiguration der Instrumentierung wird in einem Instrumentierungschema, kurz IScheme, zusammengefasst und für das entsprechende Projekt gesichert. Für alle Projekte gibt es die drei grundlegenden ISchemes, die der Instrumentierung nach den entsprechenden Leveln entsprechen, von SOTA vorgegeben.

Level 0

Die Zuweisung des Level 0 als Instrumentierungslevel für eine Struktur hat zur Folge, dass diese von der Instrumentierung ausgenommen wird. Dies ist z.B. sinnvoll, wenn man Funktionen, die viel Loginformationen (durch häufiges Ausführen oder komplexe Funktionsabläufe) erzeugen würden, aber schon hinreichend getestet worden sind, von weiteren Tests ausschließen möchte.

Level 1

Die grundlegendste Instrumentierung wird mit Level 1 angeboten. Hier werden der Funktionseingang, seine Ausgänge und sämtliche verzweigende Programmstrukturen so instrumentiert, dass aus diesen Informationen der Kontrollfluss durch die Funktion rekonstruiert werden kann. Mit diesen Daten ist es möglich alle Überdeckungsmaße bis auf die der Bedingungsüberdeckung zu berechnen.

Level 2

Zusätzlich zu Level 1 wird bei der Instrumentierung nach Level 2 auch die Belegung jedes Atoms, sofern es auch im Programm ausgewertet werden würde, in der Logdatei gesichert. Diese Daten ermöglichen es, zusätzlich zu den Maßen nach Level 1 auch die Bedingungsüberdeckungsmaße für den Programmtest berechnen zu lassen.

Level 3

Schließlich bietet SOTA mit der Instrumentierung nach Level 3 eine vollständige Instrumentierung

des Quellcodes an. Es wir hier neben den ausgewerteten Atomen auch die Ausführung sämtlicher einzelnen Anweisungen im Logfile vermerkt, wodurch dieses im Vergleich zu den der anderen Instrumentierungslevel in einem erheblichen Maße an Umfang zunehmen kann. Diese Option der Instrumentierung wird nicht nur der Vollständigkeit halber angeboten, sie ermöglicht außerdem die detaillierte Auswertung des Kontrollflusses bei unnormal terminierenden Programmen und bei der Ausnahmebehandlung.

7.4 Weitere Begriffe (Glossar)

Ant/Ant-Buildfile	Apache Ant ist ein mit *make*-vergleichbares Werkzeug zum automatischen Kompilieren von Quellprojekten, welches in der Java-Entwicklung sehr verbreitet ist. Die Ziele und Anweisungen für die Kompilation stehen in einer XML-Datei, dem Ant-Buildfile, welches von ant gelesen die Kompiliation ermöglicht. Bei der Nutzung von Eclipse besteht die Möglichkeit, sehr einfach ein ant-Buildfile über *File -> Export -> Ant Buildfile* zu exportieren.
ASC-Logger.ini / ASCLogger.jar	Für den Programmtest von Java-Programmen wird eine Logging-Komponente namens ASCLogger.jar benötigt, die das Projekt eingebunden werden muss und während des Testlaufes die Sicherung der Überdeckungsdaten übernimmt. Die Einbindung in Eclipse funktioniert dabei über *Project -> Properties -> Java Build Path -> Add JARs* bzw. *Add External JARs*, je nachdem, ob man die ASCLogger-Bibliothek in das Projekt eingefügt hat oder sie aus dem SOTA-Verzeichnis lädt. Die Informationen zum konkreten Testfall, d.h. Projektname,Testname, Beschreibung und das verwendete IScheme werden über eine Initialisierungsdatei namens ASCLogger.ini bereitgestellt, die beim Teststart im Ausführungsverzeichnis des Testprogrammes erstellt wird und von dort durch den ASCLogger gelesen wird.
Ausführungsverzeichnis des Testprogrammes	Das Ausführungsverzeichnis (execution directory) des Testprogrammes ist das Verzeichnis, aus welchem es gestartet wird. D.h. das Verzeichnis, wo man java -cp .. classname ausführt, bzw. bei der Verwendung eines Startskripts das diese Batch-Datei enthaltene Verzeichnis. Bei der RCP-Entwicklung unter Eclipse wird das RCP-Programm im Basisverzeichnis der Platform, d.h. von Eclipse, gestartet. In diesem Fall ist das Ausführungsverzeichnis also "..\eclipse\". Dieses Ausführungsverzeichnis wird benötigt, um dort die ASCLogger.ini zu erstellen, welche Informationen zum Test für die Loggingkomponente enhält, und hierhin werden auch die Logdateien geschrieben.
Basisverzeichnis des Testprogrammes	Das Basisverzeichnis des Testprogrammes ist sein Wurzelverzeichnis, wo also die Quelldateien bzw. Binaries (gegebenenfalls in Unterverzeichnissen) liegen. Von hier werden die Quellen des Projektes importiert und hierhin wird der Überdeckungsreport geschrieben.

Basisverzeichnis von SOTA	Das Basisverzeichnis von SOTA ist "..\SOTA\", wo sich die ausführbare SOTA.exe und die ASCLogger.jar-Bibliothek befindet. An diesem Ort wird auch die Projektdatei <projectname>.project sowie die Logdatei von SOTA mit den Programmausschriften erstellt.
Dynamischer Programmtest	Jeder Test eines Programmes, der seine Ausführung benötigt, ist ein dynamischer Programmtest. Dazu gehören unter anderem funktionale (Black-Box-) und strukturorientierte (White- oder Glass-Box-)Tests. SOTA dient als Werkzeug für den strukturorientierten Test dazu, für diesen die neun verschiedenen Überdeckungsmaße zu ermitteln.
Instrumentationsschema / IScheme	SOTA bietet verschiedene Level der Instrumentierung an, um eine Begrenzung des Overhead durch die Instrumentierung zu ermöglichen. Ein Instrumentationsschema (kurz: IScheme) kapselt die Informationen für eine spezifische Art der Instrumentierung des Projektes, d.h. liefert eine Zuordnung aller Funktionen des Projektes auf ein Instrumentationslevel. SOTA beinhaltet immer die drei grundlegenden ISchemes, die eine Überdeckung nach den Leveln 1, 2 und 3 ermöglichen. Wird ein neues IScheme erstellt, so werden diese Daten in der Projektdatei <projectname>.project im SOTA-Basisverzeichnis gespeichert und können in Zukunft für dieses Projekt genutzt werden.
Startskript / Batchdatei	Das Startskript (unter Windows eine Batchdatei) ist eine Datei, deren Ausführung das Starten des Testprogrammes verursacht. Es muss also lediglich den typischen Java-Aufruf "java -cp .. classname" in einer für das Projekt spezifizierten Variante enthalten. Wurde das Startskript in SOTA eingebunden, so lässt sich das Testprogramm im Manuellen Programmtest aus SOTA heraus starten.
Statische Programmanalyse	Im Gegensatz zum dynamischen Programmtest arbeitet die statische Programmanalyse ohne Ausführung des Testprogrammes, sondern lediglich mit den Informationen, die durch das Parsen des Programmes gegeben werden. SOTA ermittelt aus dem Quellcode zehn verschiedene statische Maße, die Informationen über die Struktur und Komplexität des Programmes bzw. seiner Komponenten liefern.

```
<!ELEMENT Project (Name, Language, Prefix, BackupExtension, ProjectDir, ExecDir, SourceFiles, ISchemes?,
AntLocation?, AntBuildFile?, RunScript? )>
<!ELEMENT SourceFiles (File*)>
<!ELEMENT ISchemes (IScheme*)>
<!ELEMENT IScheme (Name, Description?, Level0?, Level1?, Level2?, Level3?)>
<!ELEMENT Level0 (File*, Class*, Function*)>
<!ELEMENT Level1 (File*, Class*, Function*)>
<!ELEMENT Level2 (File*, Class*, Function*)>
<!ELEMENT Level3 (File*, Class*, Function*)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Language (#PCDATA)>
<!ELEMENT File (#PCDATA)>
<!ELEMENT Class (#PCDATA)>
<!ELEMENT Function (#PCDATA)>
<!ELEMENT Prefix (#PCDATA)>
<!ELEMENT BackupExtension (#PCDATA)>
<!ELEMENT ProjectDir (#PCDATA)>
<!ELEMENT ExecDir (#PCDATA)>
<!ELEMENT AntLocation (#PCDATA)>
<!ELEMENT AntBuildFile (#PCDATA)>
<!ELEMENT RunScript (#PCDATA)>
```

SOTA Coverage Report

Project: husemorg

created: 2009-03-23 14:14:49

Function Entry-Exit Coverage (FEEC)	35,46%
Statement Coverage (C0)	37,99%
Decision Coverage (C1)	26,29%
Condition Coverage (C2)	20,42%
Minimal Multiple Decision Coverage (MMDC)	20,12%
Modified Condition Decision Coverage (MCDC)	8,67%
Multiple Condition Coverage (C3)	20,21%
Modified Boundary-Interior Path Coverage (ModBI)	0,00%
Boundary-Interior Path Coverage (BI)	0,00%

# Files	77
# Classes (TopLevel- + inner Classes)	420 (77 + 343)
# Functions	1379
# Lines	31915
# Statements	10747
# Conditions	1178

Tests

1 Level 1 instrumentation	
2 Level 2 instrumentation	
husemorg test 1 Level 2 instrumentation	

Coverage of Classes

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Project husemorg	35,46	37,99	26,29	20,42	20,12	8,67	20,21	0,00	0,00
Class semorg.sql.tables.AbstractTable	58,06	37,93	35,29	39,47	39,47	31,58	39,47	2,13	0,02
Class semorg.sql.tables.Associate	56,25	67,71	37,50	20,00	20,00	20,00	20,00	40,00	40,00
Class semorg.gui.list.AssociateListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
Class semorg.gui.provider.AssociateTableProvider	86,05	90,00	89,19	75,00	75,00	50,00	75,00	86,49	86,49
Class semorg.gui.AssociateWindow	96,00	90,84	70,24	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.util.AssociationTabControl	41,51	23,62	11,07	11,00	9,38	4,00	11,22	2,73	1,84
Class semorg.sql.tables.Booking	52,38	58,95	30,77	26,92	26,92	0,00	26,92	9,40	9,40
Class semorg.gui.util.CalendarControl	37,84	50,00	35,00	22,22	20,00	0,00	22,22	30,30	30,30
Class semorg.sql.tables.Client	48,78	52,67	25,00	29,17	29,17	16,67	29,17	29,27	26,83
Class semorg.sql.tables.ClientBooking	17,02	18,18	6,67	4,55	4,55	0,00	4,55	10,00	10,00
Class semorg.gui.list.ClientBookingListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
Class semorg.gui.provider.ClientBookingTableProvider	19,35	0,00	0,00	0,00	0,00	0,00	0,00	12,00	12,00
Class semorg.gui.ClientBookingWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.list.ClientListWindow	46,15	40,00	33,33	50,00	50,00	0,00	50,00	35,29	35,29
Class semorg.gui.provider.ClientTableProvider	87,50	91,94	91,18	80,00	80,00	60,00	80,00	88,24	88,24
Class semorg.gui.ClientWindow	62,96	84,50	51,25	55,88	57,69	17,65	55,22	0,00	0,00
Class semorg.sql.tables.Company	91,22	84,64	43,33	46,30	46,30	7,41	46,30	6,17	6,17
Class semorg.sql.tables.CompanyBooking	47,06	47,42	18,75	20,00	20,00	20,00	20,00	38,46	38,46
Class semorg.gui.list.CompanyBookingListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
Class semorg.gui.provider.CompanyBookingTableProvider	65,52	75,00	68,00	50,00	50,00	0,00	50,00	60,87	60,87
Class semorg.gui.CompanyBookingWindow	73,91	86,23	55,56	66,67	68,42	40,00	65,52	13,16	13,16
Class semorg.sql.tables.CompanyInternalPresentation	40,00	37,82	18,75	19,23	19,23	7,69	19,23	6,80	6,80

Class semorg.gui.list.CompanyInternalPresentationListWindow	61,54	55,56	33,33	50,00	50,00	0,00	50,00	47,06	47,06
Class semorg.gui.provider.CompanyInternalPresentationTableProvider	71,43	85,71	77,78	66,67	66,67	33,33	66,67	66,67	66,67
Class semorg.gui.CompanyInternalPresentationWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.list.CompanyListWindow	50,00	40,00	33,33	50,00	50,00	0,00	50,00	35,29	35,29
Class semorg.gui.provider.CompanyTableProvider	83,67	90,00	86,05	50,00	50,00	0,00	50,00	83,72	83,72
Class semorg.gui.CompanyWindow	82,61	91,39	56,48	58,51	59,62	17,02	58,06	0,00	0,00
Class semorg.sql.tables.Conduct	0,00	0,00	---	---	---	---	---	0,00	0,00
Class semorg.gui.util.Configuration	40,00	19,23	11,11	0,00	0,00	0,00	0,00	20,00	20,00
Class semorg.gui.util.ConfigureListDlg	0,00	0,00	0,00	---	---	---	---	0,00	0,00
Class semorg.gui.util.ConfigureListWidget	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.sql.access.DBAccess	66,67	18,75	50,00	---	---	---	---	57,14	57,14
Class semorg.sql.util.DBColumn	100,00	100,00	---	---	---	---	---	100,00	100,00
Class semorg.sql.util.DBConstraint	80,00	87,50	---	---	---	---	---	80,00	80,00
Class semorg.sql.util.DBTableChangedListener	100,00	100,00	---	---	---	---	---	100,00	100,00
Class semorg.gui.util.DelayedPaymentItem	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.provider.DelayedPaymentItemTableProvider	40,00	0,00	0,00	0,00	0,00	0,00	0,00	30,00	30,00
Class semorg.gui.util.DelayedPaymentViewerControl	66,67	76,74	50,00	62,50	62,50	50,00	62,50	26,67	26,67
Class semorg.sql.util.DistinctVector	85,71	66,67	50,00	50,00	50,00	0,00	50,00	75,00	75,00
Class semorg.gui.util.EditorManager	0,00	0,00	---	---	---	---	---	0,00	0,00
Class semorg.gui.provider.EnumDlg	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.sql.tables.Enumeration	54,05	33,01	23,33	33,33	33,33	33,33	33,33	18,57	7,43
Class semorg.gui.EnumerationWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.util.ExtensibleSearchControl	42,86	61,59	34,38	34,38	34,85	25,00	34,38	5,08	2,75
Class semorg.gui.util.ForeignKeyConstraintSelector	41,18	44,30	21,05	35,71	35,71	28,57	35,71	24,39	24,39
Class semorg.sql.tables.Instructor	0,00	0,00	---	---	---	---	---	0,00	0,00
Class semorg.gui.util.ItemListener	---	---	---	---	---	---	---	---	---
Class semorg.sql.util.KeyPair	0,00	0,00	---	---	---	---	---	0,00	0,00
Class semorg.sql.tables.Lecturer	36,84	37,98	25,00	27,27	27,27	27,27	27,27	17,31	13,46
Class semorg.gui.list.LecturerListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
Class semorg.gui.provider.LecturerTableProvider	87,80	92,19	91,67	83,33	83,33	66,67	83,33	88,57	88,57
Class semorg.gui.LecturerWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.list.ListWindow	80,00	85,83	100,00	100,00	100,00	100,00	100,00	66,67	60,00
Class semorg.gui.MainWindow	85,71	98,05	83,33	75,00	78,57	50,00	71,43	71,43	42,86
Class semorg.gui.util.Messages	50,00	83,33	75,00	50,00	50,00	0,00	50,00	50,00	50,00
Class semorg.sql.tables.Person	47,83	54,29	26,67	26,67	26,67	0,00	26,67	12,36	12,36
Class semorg.sql.tables.Presentation	36,19	25,19	6,67	3,70	3,70	0,00	3,70	1,56	1,56
Class semorg.gui.list.PresentationListWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.provider.PresentationTableProvider	71,05	81,36	72,97	50,00	50,00	0,00	50,00	66,67	66,67
Class semorg.sql.tables.PublicPresentation	22,86	15,04	7,69	5,00	5,00	0,00	5,00	8,51	8,51
Class semorg.gui.list.PublicPresentationListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
Class semorg.gui.provider.PublicPresentationTableProvider	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.PublicPresentationWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.SelectLanguageWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.app.SemOrg	100,00	53,85	33,33	---	---	---	---	60,00	60,00
Class semorg.app.SemOrgStarter	100,00	40,00	50,00	---	---	---	---	50,00	50,00
Class semorg.sql.tables.SeminarType	42,70	38,18	16,67	16,67	16,67	4,76	16,67	5,11	5,11
Class semorg.gui.list.SeminarTypeListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
Class semorg.gui.provider.SeminarTypeTableProvider	73,53	82,00	75,86	50,00	50,00	0,00	50,00	71,43	71,43
Class semorg.gui.SeminarTypeWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.sql.tables.SimpleIDKey	---	---	---	---	---	---	---	---	---

Class semorg.sql.tables.Supervisor	0,00	0,00	---	---	---	---	---	0,00	0,00
Class semorg.gui.util.TableColumnProperty	0,00	0,00	---	---	---	---	---	0,00	0,00
Class semorg.gui.util.TimeControl	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Class semorg.gui.util.UtilityGUI	100,00	92,00	83,33	63,04	66,13	43,48	65,79	21,43	21,43
Class semorg.sql.util.UtilitySQL	38,46	27,03	16,67	25,00	25,00	0,00	18,75	26,32	26,32

Detailed Coverage

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.AbstractTable	58,06	37,93	35,29	39,47	39,47	31,58	39,47	2,13	0,02
- AbstractTable (Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCreationDate ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getModificationDate ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setModificationDate (Timestamp)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createExtendedQueryString (String, Vector, String)	100,00	40,68	36,36	42,31	42,31	30,77	42,31	0,44	0,00
- getColumns (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK (String)	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- getNext (String, DBColumn, int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (String, DBColumn, int)	40,00	29,41	33,33	0,00	0,00	0,00	0,00	25,00	25,00
- fireTableChangedEvent (int)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	71,43	100,00
- addDBTableChangedListener (DBTableChangedListener)	100,00	100,00	---	---	---	---	---	100,00	100,00
- removeDBTableChangedListener (DBTableChangedListener)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Associate	56,25	67,71	37,50	20,00	20,00	20,00	20,00	40,00	40,00
- Associate (int, String, String, String, String, String, String, String, String, String, String, String, String, String, Date, Date, int, String, String, String, String, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getVector (ResultSet)	100,00	93,33	75,00	100,00	100,00	100,00	100,00	50,00	50,00
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createAssociateTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getEntitlement ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getOccupation ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getPassword ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setEntitlement (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setOccupation (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setPassword (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- updateDB ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	100,00	83,33	50,00	0,00	0,00	0,00	0,00	25,00	25,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.AssociateListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
- AssociateListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- AssociateListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- AssociateListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00

- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	50,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTableInputFromDB ()	100,00	36,36	33,33	50,00	50,00	0,00	50,00	16,67	16,67
- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.AssociateTableProvider	86,05	90,00	89,19	75,00	75,00	50,00	75,00	86,49	86,49
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- getColumnImage (Object, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnText (Object, int)	87,88	91,18	89,19	75,00	75,00	50,00	75,00	87,50	87,50
- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.AssociateWindow	96,00	90,84	70,24	0,00	0,00	0,00	0,00	0,00	0,00
- AssociateWindow (Shell)	100,00	100,00	100,00	0,00	0,00	0,00	0,00	66,67	33,33
- AssociateWindow (Shell, Associate)	100,00	100,00	---	---	---	---	---	100,00	100,00
- setInput (Associate)	100,00	84,26	63,46	0,00	0,00	0,00	0,00	0,00	0,00
- setInputComponentsEnabled (boolean)	100,00	100,00	100,00	0,00	0,00	0,00	0,00	66,67	33,33
- getEntitlementValue ()	100,00	100,00	100,00	0,00	0,00	0,00	0,00	28,57	28,57
- checkInput ()	100,00	100,00	100,00	0,00	0,00	0,00	0,00	18,75	18,75
- createButtonListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- commitInputToDB ()	100,00	23,53	33,33	0,00	0,00	0,00	0,00	25,00	25,00
- confirmClose ()	80,00	76,92	66,67	0,00	0,00	0,00	0,00	37,50	37,50
- onSave ()	100,00	100,00	100,00	0,00	0,00	0,00	0,00	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.AssociationTabControl	41,51	23,62	11,07	11,00	9,38	4,00	11,22	2,73	1,84
- AssociationTabControl (Composite, int, Shell)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	66,67	33,33
- hasLockedId (SimpleIDKey)	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- onDelete ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- determineDeleteMessage (DistinctVector)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- openEditItemWindow ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- openNewItemWindow ()	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- openListWindow ()	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- onDisconnect ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- configureClassSpecificViewerParts ()	100,00	45,16	33,33	---	---	---	---	23,08	23,08
- setInput (int)	66,67	38,00	38,89	75,00	75,00	50,00	75,00	14,29	14,29
- getObjects ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setEnabled (boolean)	100,00	100,00	---	---	---	---	---	100,00	100,00
- addSelectedIds (SimpleIDKey[], boolean)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- commitIntoDB (int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- disconnectInDB ()	100,00	25,32	18,52	7,50	5,00	0,00	7,50	4,05	3,49

- connectInDB ()	100,00	2,82	3,57	25,00	16,67	0,00	25,00	0,24	0,24
- deleteFromDB ()	100,00	3,85	7,14	50,00	50,00	0,00	50,00	7,14	7,14
- setData (String, String)	100,00	100,00	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- addItemListener (ItemListener)	100,00	100,00	---	---	---	---	---	100,00	100,00
- removeItemListener (ItemListener)	0,00	0,00	---	---	---	---	---	0,00	0,00
- fireItemChanged ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- addUpdateListener ()	100,00	46,15	33,33	---	---	---	---	23,08	23,08
- update ()	0,00	0,00	0,00	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Booking	52,38	58,95	30,77	26,92	26,92	0,00	26,92	9,40	9,40
- Booking (int, Date, Date, Date, Date, Date, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createBookingTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	66,67	78,57	50,00	50,00	50,00	0,00	50,00	1,56	1,56
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- removeFromDB (Iterable)	100,00	84,62	50,00	0,00	0,00	0,00	0,00	33,33	33,33
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getBilled ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setBilled (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getConfirmed ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setConfirmed (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getEnrolled ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setEnrolled (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getMessaged ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setMessaged (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getSignedOff ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setSignedOff (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getId ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- hashCode ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- equals (Object)	66,67	66,67	50,00	50,00	50,00	0,00	50,00	50,00	50,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.CalendarControl	37,84	50,00	35,00	22,22	20,00	0,00	22,22	30,30	30,30
- CalendarControl (Composite, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- onOpen ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- fireSWTCalendarPopupClosed ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- fireSWTCalendarPopupOpened ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setActivated (boolean)	100,00	100,00	100,00	50,00	50,00	0,00	50,00	100,00	100,00
- getDate ()	66,67	66,67	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- setFont (Font)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setDate (Date)	100,00	100,00	100,00	50,00	50,00	0,00	50,00	75,00	75,00
- addSWTCalendarPopuplistener (SWTCalendarPopupListener)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeSWTCalendarPopuplistener (SWTCalendarPopupListener)	0,00	0,00	---	---	---	---	---	0,00	0,00
- addSWTCalendarlistener (SWTCalendarListener)	100,00	100,00	---	---	---	---	---	100,00	100,00
- removeSWTCalendarlistener (SWTCalendarListener)	0,00	0,00	---	---	---	---	---	0,00	0,00
- fireSWTCalendarDateChanged ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setEnabled (boolean)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getOpenPopupButton ()	0,00	0,00	---	---	---	---	---	0,00	0,00

- isPopupOpen ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- closePopup ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setToolTipText (String)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Client	48,78	52,67	25,00	29,17	29,17	16,67	29,17	29,27	26,83
- Client (int, String, String, String, String, String, String, String, String, String, String, String, String, String, Date, Date, String, Float, int, String, String, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getEmployerDescription ()	40,00	15,38	16,67	25,00	25,00	0,00	25,00	25,00	25,00
- getVector (ResultSet)	100,00	93,75	83,33	100,00	100,00	100,00	100,00	37,50	25,00
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createClientTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	100,00	85,71	50,00	50,00	50,00	0,00	50,00	25,00	25,00
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getEmployerId ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setEmployerId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTask ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setTask (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTurnover ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setTurnover (Float)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getClient (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getEmployees (int)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.ClientBooking	17,02	18,18	6,67	4,55	4,55	0,00	4,55	10,00	10,00
- ClientBooking (int, Date, Date, Date, Date, Date, int, int, int, int, Timestamp, Timestamp)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getSubstituteDescription ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getVector (ResultSet)	100,00	21,05	50,00	50,00	50,00	0,00	50,00	25,00	25,00
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createClientBookingTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getClientBooking (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getClientId ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setClientId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getDebitorId ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setDebitorId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPresentationId ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setPresentationId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getBookingsOfClient (int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getBookingsForPresentation (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getSubstituteId ()	0,00	0,00	---	---	---	---	---	0,00	0,00

- setSubstituteId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.ClientBookingListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
- ClientBookingListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- ClientBookingListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- ClientBookingListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	50,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTableInputFromDB ()	100,00	36,36	33,33	50,00	50,00	0,00	50,00	16,67	16,67
- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.ClientBookingTableProvider	19,35	0,00	0,00	0,00	0,00	0,00	0,00	12,00	12,00
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- getColumnImage (Object, int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumnText (Object, int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.ClientBookingWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- ClientBookingWindow (Shell, AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- ClientBookingWindow (Shell, ClientBooking, AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setInput (ClientBooking)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setInputComponentsEnabled (boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createButtonListener (Composite, AssociationTabControl)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- checkInput ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- commitInputToDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- confirmClose ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onSave ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.ClientListWindow	46,15	40,00	33,33	50,00	50,00	0,00	50,00	35,29	35,29
- ClientListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- ClientListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- ClientListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00

- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	50,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getTableInputFromDB ()	100,00	36,36	33,33	50,00	50,00	0,00	50,00	16,67	16,67
- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.ClientTableProvider	87,50	91,94	91,18	80,00	80,00	60,00	80,00	88,24	88,24
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- getColumnImage (Object, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnText (Object, int)	90,00	93,33	91,18	80,00	80,00	60,00	80,00	89,66	89,66
- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.ClientWindow	62,96	84,50	51,25	55,88	57,69	17,65	55,22	0,00	0,00
- ClientWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00
- ClientWindow (Shell, Client, AssociationTabControl)	100,00	100,00	---	---	---	---	---	100,00	100,00
- ClientWindow (Shell, Client, ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setInput (Client)	100,00	83,19	60,00	60,42	61,54	20,83	59,57	0,00	0,00
- setInputComponentsEnabled (boolean)	100,00	100,00	---	---	---	---	---	100,00	100,00
- createButtonListener (AssociationTabControl)	100,00	81,82	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- createButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- checkInput ()	100,00	50,00	50,00	50,00	50,00	0,00	50,00	6,25	6,25
- commitInputToDB ()	100,00	24,32	37,50	50,00	50,00	0,00	50,00	12,50	12,50
- confirmClose ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onSave ()	75,00	85,71	75,00	75,00	83,33	50,00	75,00	66,67	66,67

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Company	91,22	84,64	43,33	46,30	46,30	7,41	46,30	6,17	6,17
- Company (int, String, Date, String, String, String, Float, Date, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createCompanyTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	66,67	86,67	50,00	50,00	50,00	0,00	50,00	0,19	0,19
- updateDB ()	100,00	86,79	50,00	50,00	50,00	0,00	50,00	0,20	0,20
- removeFromDB (Iterable)	100,00	84,62	50,00	50,00	50,00	0,00	50,00	33,33	33,33
- getVector (ResultSet)	100,00	95,12	66,67	75,00	75,00	50,00	75,00	25,00	25,00
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getCompany (int)	0,00	0,00	---	---	---	---	---	0,00	0,00

- hashCode ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- equals (Object)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00
- getId ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getAnnex ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setAnnex (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCity ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCity (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCountry ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCountry (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpBirthDay ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpBirthDay (Date)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpDepartment ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpDepartment (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpEmail ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpEmail (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpFax ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpFax (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpFirstname ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpFirstname (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpMobile ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpMobile (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpName ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpName (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpPhone ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpPhone (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpSalutation ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpSalutation (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpTask ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpTask (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCpTitle ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCpTitle (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCustomerSince ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCustomerSince (Date)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getEmail ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setEmail (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getFax ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setFax (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getFirstname ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setFirstname (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getMobile ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setMobile (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getName ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setName (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getNotices ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setNotices (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getPhone ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setPhone (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getSalutation ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setSalutation (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getShortInfo ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setShortInfo (String)	100,00	100,00	---	---	---	---	---	100,00	100,00

- getShortName ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setShortName (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getStreet ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setStreet (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getTitle ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setTitle (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getTurnover ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setTurnover (Float)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getZipCode ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setZipCode (String)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.CompanyBooking	47,06	47,42	18,75	20,00	20,00	20,00	20,00	38,46	38,46
- CompanyBooking (int, Date, Date, Date, Date, Date, int, int, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getVector (ResultSet)	100,00	88,24	75,00	100,00	100,00	100,00	100,00	75,00	75,00
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createCompanyBookingTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- updateDB ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getCompanyId ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCompanyId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPresentationId ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setPresentationId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getBookingsOfCompany (int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getBookingsForPresentation (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getCompanyBooking (int)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.CompanyBookingListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
- CompanyBookingListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- CompanyBookingListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- CompanyBookingListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	50,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTableInputFromDB ()	100,00	36,36	33,33	50,00	50,00	0,00	50,00	16,67	16,67
- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI

Class semorg.gui.provider.CompanyBookingTableProvider	65,52	75,00	68,00	50,00	50,00	0,00	50,00	60,87	60,87
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- getColumnImage (Object, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnText (Object, int)	57,89	76,32	68,00	50,00	50,00	0,00	50,00	55,56	55,56
- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.CompanyBookingWindow	73,91	86,23	55,56	66,67	68,42	40,00	65,52	13,16	13,16
- CompanyBookingWindow (Shell, AssociationTabControl)	100,00	100,00	---	---	---	---	---	100,00	100,00
- CompanyBookingWindow (Shell, CompanyBooking, AssociationTabControl)	100,00	100,00	---	---	---	---	---	100,00	100,00
- setInput (CompanyBooking)	100,00	100,00	91,67	85,71	83,33	71,43	84,62	4,17	4,17
- setInputComponentsEnabled (boolean)	100,00	100,00	---	---	---	---	---	100,00	100,00
- createButtonListener (Composite, AssociationTabControl)	100,00	84,62	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- checkInput ()	100,00	54,55	50,00	50,00	50,00	0,00	50,00	12,50	12,50
- commitInputToDB ()	100,00	31,58	33,33	50,00	50,00	0,00	50,00	25,00	25,00
- confirmClose ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onSave ()	75,00	85,71	75,00	75,00	83,33	50,00	75,00	66,67	66,67

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.CompanyInternalPresentation	40,00	37,82	18,75	19,23	19,23	7,69	19,23	6,80	6,80
- CompanyInternalPresentation (int, int, Integer, Date, Date, Time, Time, Time, Time, String, String, String, String, String, String, boolean, Float, Integer, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getVector (ResultSet)	100,00	84,85	60,00	62,50	62,50	25,00	62,50	1,35	1,35
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createCompanyInternalPresentationTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnsWithSemtype ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getCompanyInternalPresentation (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getCurrentEntrantsCount ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getMaxEntrants ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setMaxEntrants (Integer)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPackagePrice ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setPackagePrice (Float)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.CompanyInternalPresentationListWindow	61,54	55,56	33,33	50,00	50,00	0,00	50,00	47,06	47,06
- CompanyInternalPresentationListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- CompanyInternalPresentationListWindow (ForeignKeyConstraintSelector)	100,00	100,00	---	---	---	---	---	100,00	100,00
- CompanyInternalPresentationListWindow	0,00	0,00	---	---	---	---	---	0,00	0,00

(AssociationTabControl)									
- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedToolBarListeners ()	50,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	100,00	100,00	---	---	---	---	---	100,00	100,00
- openList ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTableInputFromDB ()	100,00	33,33	33,33	50,00	50,00	0,00	50,00	16,67	16,67
- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.CompanyInternalPresentationTableProvider	71,43	85,71	77,78	66,67	66,67	33,33	66,67	66,67	66,67
- getColumnText (Object, int)	71,43	85,71	77,78	66,67	66,67	33,33	66,67	66,67	66,67

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.CompanyInternalPresentationWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- CompanyInternalPresentationWindow (Shell)	0,00	0,00	---	---	---	---	---	0,00	0,00
- CompanyInternalPresentationWindow (Shell, CompanyInternalPresentation, AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- CompanyInternalPresentationWindow (Shell, CompanyInternalPresentation, ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setInput (CompanyInternalPresentation)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setInputComponentsEnabled (boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createButtonListener (AssociationTabControl)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- createButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- checkInput ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- commitInputToDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- confirmClose ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onSave ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.CompanyListWindow	50,00	40,00	33,33	50,00	50,00	0,00	50,00	35,29	35,29
- CompanyListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- CompanyListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- CompanyListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	75,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getTableInputFromDB ()	100,00	36,36	33,33	50,00	50,00	0,00	50,00	16,67	16,67

- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.CompanyTableProvider	83,67	90,00	86,05	50,00	50,00	0,00	50,00	83,72	83,72
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- getColumnImage (Object, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnText (Object, int)	84,62	91,03	86,05	50,00	50,00	0,00	50,00	84,21	84,21
- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.CompanyWindow	82,61	91,39	56,48	58,51	59,62	17,02	58,06	0,00	0,00
- CompanyWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00
- CompanyWindow (Shell, Company, ForeignKeyConstraintSelector)	100,00	100,00	---	---	---	---	---	100,00	100,00
- setInput (Company)	100,00	80,26	56,94	57,14	58,11	14,29	56,52	0,00	0,00
- setInputComponentsEnabled (boolean)	100,00	100,00	---	---	---	---	---	100,00	100,00
- createButtonListener (ForeignKeyConstraintSelector)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00
- checkInput ()	100,00	43,48	50,00	50,00	50,00	0,00	50,00	0,78	0,78
- commitInputToDB ()	100,00	84,00	60,00	100,00	100,00	100,00	100,00	12,50	12,50
- confirmClose ()	40,00	46,15	33,33	50,00	50,00	0,00	50,00	12,50	12,50
- onSave ()	75,00	85,71	75,00	75,00	83,33	50,00	75,00	66,67	66,67

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Conduct	0,00	0,00	---	---	---	---	---	0,00	0,00
- Conduct (int, int, Timestamp, Timestamp)	0,00	0,00	---	---	---	---	---	0,00	0,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createConductTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- deleteFromDB ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumns ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getLecturerId ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getSeminartypeId ()	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.Configuration	40,00	19,23	11,11	0,00	0,00	0,00	0,00	20,00	20,00
- Configuration (boolean)	100,00	36,36	20,00	0,00	0,00	0,00	0,00	25,00	25,00
- saveConfiguration ()	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- makeStdConfiguration (boolean)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setProperty (String, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getProperty (String)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.ConfigureListDlg	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- ConfigureListDlg (Shell, TableColumn[], int[])	0,00	0,00	---	---	---	---	---	0,00	0,00
- createButtonsForButtonBar (Composite)	0,00	0,00	---	---	---	---	---	0,00	0,00
- buttonPressed (int)	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- okPressed ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- cancelPressed ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createDialogArea (Composite)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumnProperties ()	0,00	0,00	---	---	---	---	---	0,00	0,00

- configureShell (Shell)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.ConfigureListWidget	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- ConfigureListWidget (Composite, TableColumn[], int[])	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setListViewerInputs (TableColumnProperty[], int[])	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- updateColumnNumbers ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumns ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- onSelect ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- onDeselect ()	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.access.DBAccess	66,67	18,75	50,00	---	---	---	---	57,14	57,14
- DBAccess ()	100,00	50,00	50,00	---	---	---	---	50,00	50,00
- createDatabaseAndConnect ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createTables ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- init ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getPreparedStatement (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getStatement ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.util.DBColumn	100,00	100,00	---	---	---	---	---	100,00	100,00
- DBColumn (String, String, Class)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getInternalColumnName ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getPublicColumnName ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnType ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.util.DBConstraint	80,00	87,50	---	---	---	---	---	80,00	80,00
- DBConstraint (String, int, Object, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnName ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getConjunction ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getRelation ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getValue ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.util.DBTableChangedListener	100,00	100,00	---	---	---	---	---	100,00	100,00
- DBTableChangedListener (int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getType ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.DelayedPaymentItem	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- DelayedPaymentItem (int, Date)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPresentationId ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getBilled ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getDebt ()	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.DelayedPaymentItemTableProvider	40,00	0,00	0,00	0,00	0,00	0,00	0,00	30,00	30,00
- getColumnImage (Object, int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumnText (Object, int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.DelayedPaymentViewerControl	66,67	76,74	50,00	62,50	62,50	50,00	62,50	26,67	26,67
- DelayedPaymentViewerControl (Composite, Shell)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	33,33	33,33
- setInput (int)	66,67	60,00	50,00	50,00	50,00	33,33	50,00	22,22	22,22
- addUpdateListener (int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- update ()	0,00	0,00	0,00	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.util.DistinctVector	85,71	66,67	50,00	50,00	50,00	0,00	50,00	75,00	75,00
- DistinctVector ()	100,00	---	---	---	---	---	---	100,00	100,00
- DistinctVector (Collection)	100,00	---	---	---	---	---	---	100,00	100,00
- add (E)	66,67	66,67	50,00	50,00	50,00	0,00	50,00	50,00	50,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.EditorManager	0,00	0,00	---	---	---	---	---	0,00	0,00
- addEditingId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeEditingId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- isEditing (int)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.EnumDlg	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- EnumDlg (Shell, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createButtonsForButtonBar (Composite)	0,00	0,00	---	---	---	---	---	0,00	0,00
- buttonPressed (int)	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- okPressed ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- cancelPressed ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createDialogArea (Composite)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getNewValue ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- configureShell (Shell)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Enumeration	54,05	33,01	23,33	33,33	33,33	33,33	33,33	18,57	7,43
- Enumeration (int, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- createEnumerationTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getVectorFromDB (int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getVector (ResultSet)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	66,67	33,33
- getValue ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getType ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createNewEnumInDB (int, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- removeFromDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setValueInDB (String)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- addEnums (Combo, int)	100,00	83,33	75,00	100,00	100,00	100,00	100,00	50,00	25,00
- addSalutationEnums (Combo)	100,00	100,00	---	---	---	---	---	100,00	100,00
- addTitleEnums (Combo)	100,00	100,00	---	---	---	---	---	100,00	100,00
- addCountryEnums (Combo)	100,00	100,00	---	---	---	---	---	100,00	100,00
- addAnnexEnums (Combo)	100,00	100,00	---	---	---	---	---	100,00	100,00

- insertSalutationStringInDB (String)	100,00	100,00	100,00	---	---	---	---	100,00	100,00
- insertTitleStringInDB (String)	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- insertcountryStringInDB (String)	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- insertAnnexStringInDB (String)	0,00	0,00	0,00	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.EnumerationWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- EnumerationWindow (Shell)	0,00	0,00	---	---	---	---	---	0,00	0,00
- onDelete (Enumeration)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onNew ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onEdit (Enumeration)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.ExtensibleSearchControl	42,86	61,59	34,38	34,38	34,85	25,00	34,38	5,08	2,75
- ExtensibleSearchControl (Composite, int)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	66,67	33,33
- createFilterExtension (int)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00
- addExtensionListener (SelectionListener)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	66,67	33,33
- addModifyListener (ModifyListener)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getConstraints ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setColumns (Vector)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	55,56	4,76
- getComparatorString (String)	33,33	50,00	42,86	42,86	42,86	14,29	42,86	25,00	25,00
- getDBColumn (String)	100,00	100,00	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- setData (String, String)	100,00	100,00	75,00	75,00	75,00	50,00	75,00	50,00	25,00
- getConjunction (String)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getRelation (String)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.ForeignKeyConstraintSelector	41,18	44,30	21,05	35,71	35,71	28,57	35,71	24,39	24,39
- ForeignKeyConstraintSelector (Composite, int)	0,00	---	---	---	---	---	---	0,00	0,00
- ForeignKeyConstraintSelector (Composite, int, boolean)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00
- updateText ()	100,00	44,74	55,56	100,00	100,00	100,00	100,00	37,50	37,50
- setSelectedId (int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- addModifyListener (ModifyListener)	100,00	100,00	---	---	---	---	---	100,00	100,00
- setEnabled (boolean)	100,00	100,00	---	---	---	---	---	100,00	100,00
- setData (String, String)	100,00	100,00	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- isLockedId (int)	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- onDelete ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- determineDeleteMessage ()	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- deleteFromDB ()	0,00	0,00	0,00	---	---	---	---	0,00	0,00
- getSelectedId ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Instructor	0,00	0,00	---	---	---	---	---	0,00	0,00
- Instructor (int, int, Timestamp, Timestamp)	0,00	0,00	---	---	---	---	---	0,00	0,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createInstructorTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- deleteFromDB ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumns ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getLecturerId ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPresentationId ()	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.ItemListener	---	---	---	---	---	---	---	---	---

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.util.KeyPair	0,00	0,00	---	---	---	---	---	0,00	0,00
- KeyPair (int, int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- equals (KeyPair)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getKey1 ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getKey2 ()	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Lecturer	36,84	37,98	25,00	27,27	27,27	27,27	27,27	17,31	13,46
- Lecturer (int, String, String, String, String, String, String, String, String, String, String, String, String, String, Date, Date, String, Float, Float, String, String, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getVector (ResultSet)	100,00	94,12	87,50	100,00	100,00	100,00	100,00	13,64	4,55
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createLecturerTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getBio ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setBio (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getDailyFee ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setDailyFee (Float)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getHourlyFee ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setHourlyFee (Float)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getLectInstructing (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getLectSupervising (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getLectConducting (int)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.LecturerListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
- LecturerListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- LecturerListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- LecturerListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	50,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTableInputFromDB ()	100,00	36,36	33,33	50,00	50,00	0,00	50,00	16,67	16,67
- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.LecturerTableProvider	87,80	92,19	91,67	83,33	83,33	66,67	83,33	88,57	88,57
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- getColumnImage (Object, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnText (Object, int)	90,32	93,55	91,67	83,33	83,33	66,67	83,33	90,00	90,00
- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.LecturerWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- LecturerWindow (Shell, AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- LecturerWindow (Shell, Lecturer, AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setInput (Lecturer)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setInputComponentsEnabled (boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createButtonListener (Composite, AssociationTabControl)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- checkInput ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- commitInputToDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- confirmClose ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onSave ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.ListWindow	80,00	85,83	100,00	100,00	100,00	100,00	100,00	66,67	60,00
- ListWindow (Shell, Vector)	100,00	100,00	---	---	---	---	---	100,00	100,00
- ListWindow (Shell, Vector, AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- ListWindow (Shell, Vector, ForeignKeyConstraintSelector)	100,00	100,00	---	---	---	---	---	100,00	100,00
- createButtonArea ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setActive ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setVisible (boolean)	100,00	100,00	---	---	---	---	---	100,00	100,00
- update ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createListToolbarBasics (boolean)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	50,00	50,00
- createTableBasics ()	100,00	100,00	100,00	100,00	100,00	100,00	100,00	66,67	33,33
- addUpdateListener (int)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.MainWindow	85,71	98,05	83,33	75,00	78,57	50,00	71,43	71,43	42,86
- MainWindow (Display)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	71,43	14,29
- createListWindows ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- centerShell ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createStatusBar ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createMenu ()	100,00	99,17	50,00	50,00	50,00	0,00	33,33	50,00	50,00
- getShell ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getInstance ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.Messages	50,00	83,33	75,00	50,00	50,00	0,00	50,00	50,00	50,00
- Messages ()	0,00	---	---	---	---	---	---	0,00	0,00
- getString (String)	75,00	83,33	75,00	50,00	50,00	0,00	50,00	66,67	66,67

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Person	47,83	54,29	26,67	26,67	26,67	0,00	26,67	12,36	12,36
- Person (int, String, String, String, String, String, String, String, String, String, String, String, String, String, Date, Date, String, String, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getAnnex ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getBirthday ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCity ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCountry ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getEmail ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getFax ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getFirstname ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getMobile ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getName ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getNotices ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getId ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getPhone ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getSalutation ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getShortinfo ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getStreet ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getTitle ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getZipCode ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getFirstContact ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setAnnex (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setBirthday (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setCity (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setCountry (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setEmail (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setFax (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setFirstContact (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setFirstname (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setMobile (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setName (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setNotices (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setNumber (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setPhone (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setSalutation (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setShortinfo (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setStreet (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setTitle (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setZipCode (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createPersonTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	66,67	85,71	50,00	50,00	50,00	0,00	50,00	1,47	1,47
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- removeFromDB (Iterable)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- hashCode ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- equals (Object)	66,67	66,67	50,00	50,00	50,00	0,00	50,00	50,00	50,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI

Class semorg.sql.tables.Presentation	36,19	25,19	6,67	3,70	3,70	0,00	3,70	1,56	1,56
- Presentation (int, int, Integer, Date, Date, Time, Time, Time, Time, String, String, String, String, String, String, boolean, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getSeminarTypeDescription ()	50,00	70,00	50,00	50,00	50,00	0,00	50,00	33,33	33,33
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createPresentationTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- removeFromDB (Iterable)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getVector (ResultSet)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPresSupervisedBy (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPresInstructedBy (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPresForSemType (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPresentationVectorFromDB (Vector, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPresentation (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumnsWithSemtype ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- hashCode ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- equals (Object)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getShortDescription ()	100,00	87,50	50,00	50,00	50,00	0,00	50,00	25,00	25,00
- getSeminarTypeForPresId (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getAnnex ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setAnnex (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getBeginningDay ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setBeginningDay (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- isCancelled ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCancelled (boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getCity ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCity (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getCountry ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCountry (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getDuration ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setDuration (Integer)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getEndingDay ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setEndingDay (Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getEndingTime ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setEndingTime (Time)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getFirstStartingTime ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setFirstStartingTime (Time)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getLastEndingTime ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setLastEndingTime (Time)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getLocation ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setLocation (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getSeminarTypeId ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setSeminarTypeId (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getStartingTime ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setStartingTime (Time)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getStreet ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setStreet (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getZipCode ()	100,00	100,00	---	---	---	---	---	100,00	100,00

- setZipCode (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getId ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.PresentationListWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- PresentationListWindow ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- PresentationListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- PresentationListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	0,00	---	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTableInputFromDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- createWindow (Shell)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.PresentationTableProvider	71,05	81,36	72,97	50,00	50,00	0,00	50,00	66,67	66,67
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- getColumnImage (Object, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnText (Object, int)	67,86	82,46	72,97	50,00	50,00	0,00	50,00	64,29	64,29
- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.PublicPresentation	22,86	15,04	7,69	5,00	5,00	0,00	5,00	8,51	8,51
- PublicPresentation (int, int, Integer, Date, Date, Time, Time, Time, Time, String, String, String, String, String, String, boolean, String, Float, Timestamp, Timestamp)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getVector (ResultSet)	100,00	12,90	25,00	16,67	16,67	0,00	16,67	4,55	4,55
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createPublicPresentationTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnsWithSemtype ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getCoopPartner ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setCoopPartner (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getCancelFee ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setCancelFee (Float)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPublicPresentation (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getCurrentEntrantsCount ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.PublicPresentationListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
- PublicPresentationListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- PublicPresentationListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- PublicPresentationListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	50,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTableInputFromDB ()	100,00	36,36	33,33	50,00	50,00	0,00	50,00	16,67	16,67
- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.PublicPresentationTableProvider	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getColumnText (Object, int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.PublicPresentationWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- PublicPresentationWindow (Shell)	0,00	0,00	---	---	---	---	---	0,00	0,00
- PublicPresentationWindow (Shell, PublicPresentation, AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- PublicPresentationWindow (Shell, PublicPresentation, ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setInput (PublicPresentation)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setInputComponentsEnabled (boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createButtonListener (AssociationTabControl)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- createButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- checkInput ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- commitInputToDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- confirmClose ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onSave ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.SelectLanguageWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- SelectLanguageWindow (Shell)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getCurrentLanguage ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.app.SemOrg	100,00	53,85	33,33	---	---	---	---	60,00	60,00
- initializeApplication (boolean)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getConfiguration ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- main (String[])	100,00	33,33	33,33	---	---	---	---	33,33	33,33

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.app.SemOrgStarter	100,00	40,00	50,00	---	---	---	---	50,00	50,00
- main (String[])	100,00	40,00	50,00	---	---	---	---	50,00	50,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.SeminarType	42,70	38,18	16,67	16,67	16,67	4,76	16,67	5,11	5,11
- SeminarType (int, String, String, String, String, String, String, Integer, String, String, String, Float, Integer, Integer, Timestamp, Timestamp)	100,00	100,00	---	---	---	---	---	100,00	100,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSeminarTypeTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00
- insertIntoDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- updateDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- removeFromDB (Iterable)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getVector (ResultSet)	100,00	83,87	58,33	60,00	60,00	20,00	60,00	0,36	0,36
- getVectorFromDB (Vector, String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getSeminarTypeFromDB (int)	66,67	83,33	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- getColumns ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getAudience ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setAudience (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getCharge ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setCharge (Float)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getDocuments ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setDocuments (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getDuration ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setDuration (Integer)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getMaxEntrants ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setMaxEntrants (Integer)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getMethodology ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setMethodology (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getMinEntrants ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setMinEntrants (Integer)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getObjective ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setObjective (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getRequirements ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setRequirements (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getRoutine ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setRoutine (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getShortTitle ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setShortTitle (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTitle ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setTitle (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTopic ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- setTopic (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getId ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- getNext (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getPrevious (int)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getSemTypeConductedBy (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- hashCode ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- equals (SeminarType)	0,00	0,00	---	---	---	---	---	0,00	0,00
- equals (Object)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.list.SeminarTypeListWindow	38,46	37,14	33,33	50,00	50,00	0,00	50,00	29,41	29,41
- SeminarTypeListWindow ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- SeminarTypeListWindow (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- SeminarTypeListWindow (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- createSpecializedMainTableListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedMainTableListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSpecializedToolBarListeners ()	50,00	100,00	---	---	---	---	---	100,00	100,00
- createChooseButtonListener (AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createChooseButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- openList ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getTableInputFromDB ()	100,00	36,36	33,33	50,00	50,00	0,00	50,00	16,67	16,67
- createWindow (Shell)	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.provider.SeminarTypeTableProvider	73,53	82,00	75,86	50,00	50,00	0,00	50,00	71,43	71,43
- dispose ()	100,00	---	---	---	---	---	---	100,00	100,00
- getColumnImage (Object, int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- getColumnText (Object, int)	70,83	83,33	75,86	50,00	50,00	0,00	50,00	69,57	69,57
- addListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00
- isLabelProperty (Object, String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeListener (ILabelProviderListener)	100,00	---	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.SeminarTypeWindow	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- SeminarTypeWindow (Shell)	0,00	0,00	---	---	---	---	---	0,00	0,00
- SeminarTypeWindow (Shell, SeminarType, AssociationTabControl)	0,00	0,00	---	---	---	---	---	0,00	0,00
- SeminarTypeWindow (Shell, SeminarType, ForeignKeyConstraintSelector)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setInput (SeminarType)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setInputComponentsEnabled (boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- createButtonListener (AssociationTabControl)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- createButtonListener (ForeignKeyConstraintSelector)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- checkInput ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- commitInputToDB ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- confirmClose ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- onSave ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.SimpleIDKey	---	---	---	---	---	---	---	---	---

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.sql.tables.Supervisor	0,00	0,00	---	---	---	---	---	0,00	0,00
- Supervisor (int, int, Timestamp, Timestamp)	0,00	0,00	---	---	---	---	---	0,00	0,00
- tableOK ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- createSupervisorTable (Statement)	0,00	0,00	---	---	---	---	---	0,00	0,00

- insertIntoDB ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- deleteFromDB ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumns ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getLecturerId ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- getPresentationId ()	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.TableColumnProperty	0,00	0,00	---	---	---	---	---	0,00	0,00
- TableColumnProperty (int, int, String, int, int, boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getAlignment ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setAlignment (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getColumnNumber ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setColumnNumber (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getResizable ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setResizable (boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getWidth ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setWidth (int)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getText ()	0,00	0,00	---	---	---	---	---	0,00	0,00
- setText (String)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getIndex ()	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.TimeControl	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- TimeControl (Composite)	0,00	0,00	---	---	---	---	---	0,00	0,00
- getHours ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getMinutes ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- getTime ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- fireInputChanged ()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- addTimeChangeListener (TimeChangeListener)	0,00	0,00	---	---	---	---	---	0,00	0,00
- removeTimeChangeListener (TimeChangeListener)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setTime (Time)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setActivated (boolean)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setEnabled (boolean)	0,00	0,00	---	---	---	---	---	0,00	0,00
- setData (String, String)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- setToolTipText (String)	0,00	0,00	---	---	---	---	---	0,00	0,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI
Class semorg.gui.util.UtilityGUI	100,00	92,00	83,33	63,04	66,13	43,48	65,79	21,43	21,43
- getFormData (Object, int, Object, int, Object, int, Object, int)	100,00	100,00	100,00	83,33	87,50	66,67	80,00	12,35	12,35
- isvalidPercentage (int)	100,00	100,00	---	---	---	---	---	100,00	100,00
- createTextTab (TabFolder, String, Font, int)	100,00	86,67	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- setLength (Text, int)	100,00	87,50	50,00	25,00	33,33	0,00	33,33	50,00	50,00
- setLength (Text, int, int)	100,00	87,50	50,00	25,00	33,33	0,00	33,33	50,00	50,00
- setLength (Combo, int)	100,00	87,50	50,00	25,00	33,33	0,00	33,33	50,00	50,00
- alignRight (Control)	100,00	85,71	50,00	25,00	33,33	0,00	33,33	50,00	50,00
- getImage (String)	100,00	100,00	100,00	100,00	100,00	100,00	100,00	75,00	75,00
- initImage (String)	100,00	100,00	---	---	---	---	---	100,00	100,00
- disposeImages ()	100,00	100,00	---	---	---	---	---	100,00	100,00

top	FEEC	C0	C1	C2	MMDC	MCDC	C3	ModBI	BI

Class semorg.sql.util.UtilitySQL	38,46	27,03	16,67	25,00	25,00	0,00	18,75	26,32	26,32
- convertToSQLDate (java.util.Date)	0,00	0,00	---	---	---	---	---	0,00	0,00
- convertToTimestamp (java.util.Date)	100,00	100,00	---	---	---	---	---	100,00	100,00
- parseFloat (String)	50,00	33,33	25,00	50,00	50,00	0,00	33,33	33,33	33,33
- isCorrectFloat (String)	66,67	33,33	50,00	50,00	50,00	0,00	33,33	50,00	50,00
- parseInteger (String)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- isCorrectInteger (String)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
- createFloatText (Float)	66,67	66,67	50,00	50,00	50,00	0,00	50,00	50,00	50,00
- initUtilitySQL ()	100,00	100,00	---	---	---	---	---	100,00	100,00
- castArray (Object[], Class)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00